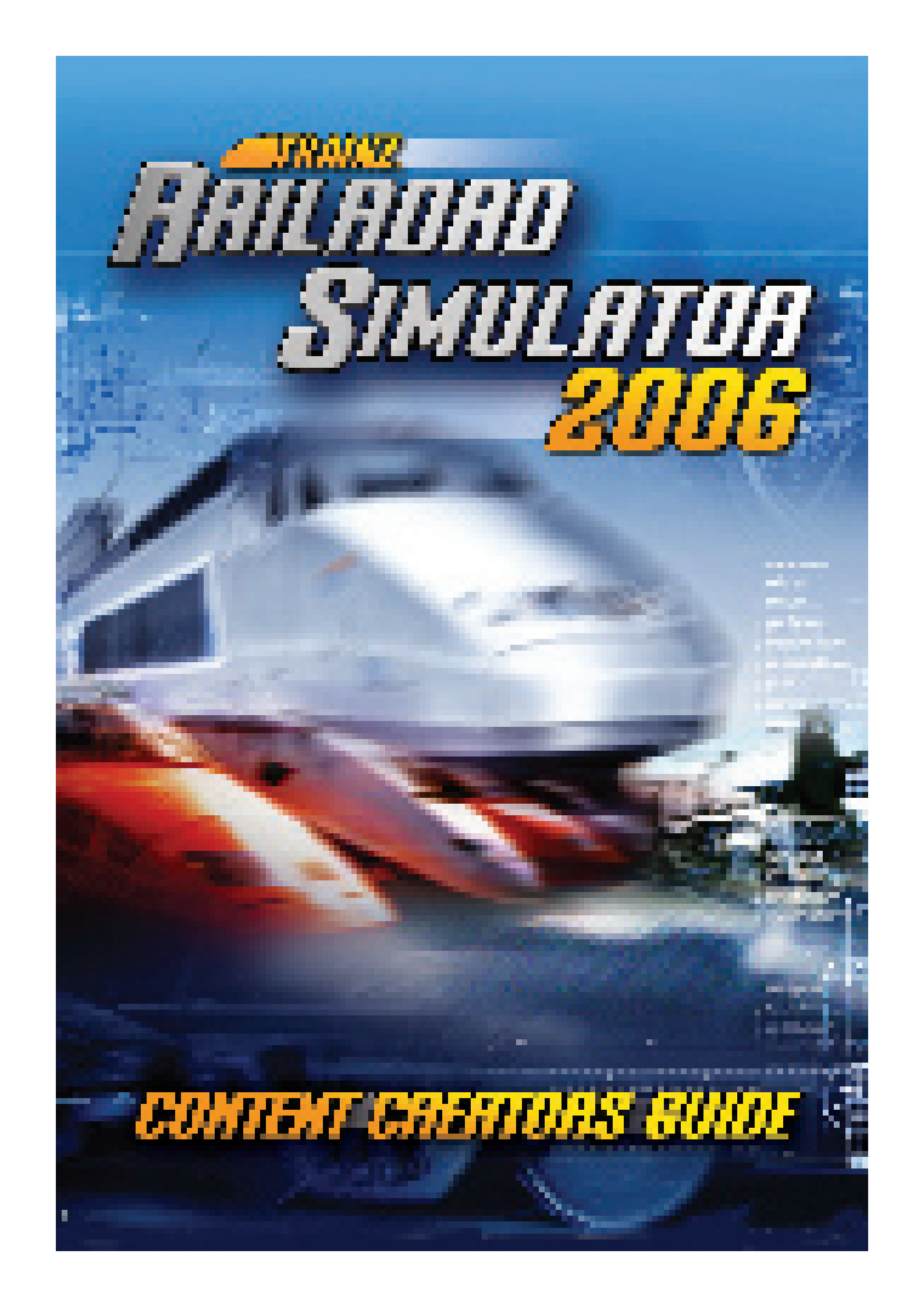




THANE **RAILROAD** **SIMULATOR** **2006**

CONTENT CREATORS GUIDE

Trainz Railroad Simulator 2006

Content Creator's Guide

Version 2
August 2006

Copyright© Auran Pty Ltd 2005

This Content Creators Guide is for the Trainz TRS 2006 release. There have been many changes to TRS2006, new options, and the introduction of the Content Managers Plus program to manage content, and the integrated Content Creator Plus module, to allow the creation of in-game content.

This document outlines the standards and procedures to create content for Trainz TRS2006, content that is suitable for uploading to the Download Station.

Further updates may be available from time to time on our website. <http://www.auran.com/TRS2006/index.php>

CONTENTS

CHAPTER 1

The Basics

INTRODUCTION	1
OVERVIEW	2
What's new in TRS2006?	2
The Basics of Content Creation	2
WHAT IS A KUID?	4
KUID2 FORMAT	4
USER ID NUMBER	4
VERSION ID NUMBER	5

CHAPTER 2

Introduction to Kinds, Containers, Tags, and Config.txt files

KINDS	7
Config.txt Files	8
Containers	8
Tags	9
Directory Structure	9
Kinds and Container Relationship	9
CONFIG.TXT FILES	10
EFFECTS (optional mesh-table variables)	12
EFFECT: KIND NAME	12
EFFECT: KIND CORONA	13
EFFECT: KIND TEXTURE-REPLACEMENT	13
EFFECT: KIND ATTACHMENT	14
EFFECT: KIND ANIMATION	15
KIND: HTML-ASSET	15

KIND: PRODUCT	16
KIND: PRODUCT-CATEGORY	16
OVERVIEW	16
IN-GAME VISUALISATION OF PRODUCTS.	16
AVIATION FUEL PRODUCT	17
COAL PRODUCT	17
GENERAL GOODS PRODUCT	18
GENERAL GOODS MESH DIMENSIONS	18
CRUDE OIL PRODUCT	19
DIESEL FUEL PRODUCT	19
20 FT CONTAINER PRODUCT	20
20 FT CONTAINER MESH DIMENSIONS	20
40 FT CONTAINER PRODUCT	21
40 FT CONTAINER MESH DIMENSIONS	21
LUMBER PRODUCT	22
LOG MESH DIMENSIONS	22
PETROL FUEL PRODUCT	22
LUMBER MESH DIMENSIONS	22
WATER PRODUCT	23
WOODCHIPS PRODUCT	23
PASSENGER PRODUCT	24
KIND: ENGINE	25
DIESEL ENGINE FILE BREAKDOWN	29
STEAM ENGINE FILE BREAKDOWN	31
COMMENTS AND NEW TAGS	31
KIND: BOGEY	33
BREAKDOWN OF CONFIG.TXT	33
KIND: TRAINCAR	34
TRAINCAR CONFIG.TXT BREAKDOWN	35
TRAINCAR EXAMPLES	36
ENGINESOUND - DIESEL AND ELECTRIC	37
KIND: ENGINESOUND	37
ENGINESOUND - STEAM	38
KIND: HORNSOUND	39
KIND: DRIVERCHARACTER	39
KIND: INTERIOR	40

INTERIOR CONFIG.TXT BREAKDOWN	47
STEAM CAB INTERIORS	50
KIND: PANTOGRAPH	62
KIND: ENVIRONMENT	62
KIND: WATER2	63
CALM WATER	63
CHOPPY WATER	64
GLASSY WATER	65
ROUGH WATER	65
KIND: MAP	66
KIND: PROFILE	66
KIND: GROUNDTEXTURE	66
GROUNDTEXTURE CLUTTER MESH	66
KIND: SCENERY	67
KIND: INDUSTRY	69
PORTAL	70
MULTI INDUSTRY NEW	72
PASSENGER STATION ASSET	74
PASSENGER VEHICLE ASSET	76
KIND: FIXEDTRACK	77
KIND: TRACK – RAILS	78
KIND: TRACK – ROAD	79
KIND: BRIDGE – SINGLE TRACK	80
KIND: BRIDGE – TUNNEL	81
KIND: BRIDGE – DOUBLE TRACK	82
KIND: MOSPEEDBOARD	82
KIND: MOSIGNAL	83
KIND: MOJUNCTION	85
KIND: MESH	85
KIND: MOTURNTABLE	86
KIND: MOCROSSING	87
KIND: ACTIVITY	88
KIND: TEXTURE	88
KIND: BUILDABLE	89

CHAPTER 3

Understanding and using Content Creator Plus

Getting Started	91
Using Content Creator Plus	93
Inheritance Template	95

CHAPTER 4

Using Content Creator Plus to create a New Asset

Creating a New Asset	97
A Workflow Process	97
Example Asset	97
Example Asset PB15 Directory Layout	98
Example Asset Main CCP Screen	100

CHAPTER 5

Containers and Tags Common to Each Kind

Introduction	103
Common Items	103
Optional and Mandatory Requirements	103
Common Containers	103
Other Common Containers	106
Mesh Table	106
Other Common Tags	107

CHAPTER 6

All Other Container and Tags

CHAPTER 7

Kind Examples

ENVIRONMENT	116
GROUNDTEXTURE	116
DISPLACEMENTS	116

SCENERY	117
SCENERY OBJECTS WITH LIGHTS AT NIGHT	117
SCENERY OBJECTS WITH ANIMATION	117
SPECIAL SCENERY OBJECTS – LEVEL CROSSINGS	118
SPECIAL SCENERY OBJECTS - TURNABLES	118
SCENERY OBJECTS WITH SOUNDS	118
SPLINES	119
TRACK	120
RAILS	120
BRIDGES	120
TRACKSIDE	121
PANTOGRAPH	121
TUNNELS	121

CHAPTER 8

Modeling Guidelines

3DSMax/gmax Interface with Trainz	123
1. 3DSMAX/GMAX INITIAL SETUP	123
2. MERGING AND EXPORTING	123
3. ANIMATION REQUIREMENTS	124
4. ATTACHMENTS	125
5. GENERAL MODELING NOTES	125
6. CONFIG.TXT FILE	126
7. PROBLEMS WITH MODEL EXPORTS	127
POLYCOUNT	128
TRAINS	130
LOCOMOTIVE NUMBERING	131
BUMP MAPPED AND SPECULAR MATERIALS	131
BUMP MAPPING INFORMATION	132
Textures and Opacity Effects	136
1. PLACEMENT IN 3DSMAX/GMAX	136
2. OPACITY FADE OUT	136
3. ALPHA CHANNEL USE	136
4. EXAMPLE	136
5. APPLYING OPACITY TO MODELS	136
6. OPACITY SETTINGS IN 3DSMAX/GMAX	137
7. OPACITY INTERFERENCE	137

8. SPECIAL USE OF OPACITY - REFLECTION MATERIALS	137
9. OPACITY ON ROADS, TRACK AND BRIDGES	138
10. OPACITY TEXTURE BLEEDING	138
11. TEXTURES FOR TILING	138
CREATING AN INTERIOR FOR TRS	139
STEAM CAB INTERIORS	143
STEAM CAB FIRE AND COAL GLOW EFFECTS	145
Researching Data and Testing of a Steam Locomotive	146
NARROW GAUGE GEARED LOCOMOTIVES	146
TENDERS	147
Content Creation Guide Amendments	147
DITCH LIGHTS	147
NAMING OF ASSETS	147
ANIMATION EVENTS	148
LEVEL OF DETAIL MESH REDUCTION	149
LOAD TEXTURE REPLACEMENT	151
TRAINCAR DIRECTORY STRUCTURE	154
ALIASING TRAINS	155
BOGEYS	156
3D STUDIO MAX AND GMAX MODEL GUIDELINES	156
PANTOGRAPHS (pants)	159

CHAPTER 9

Uploading to the Download Station

TRAINZ DOWNLOAD STATION	160
TRAINZ DOWNLOAD HELPER	162
Uploading Models and Forum Issues	163
1. PACKAGING ASSETS	163
2. PLACING SCREENSHOTS ON THE FORUM	163

CHAPTER 11

Particle Effects and Scripts

INTRODUCTION	164
MAIN PROPERTIES:	165
PFX FROM CONFIG.TXT	165
ADDING SMOKE TAGS	165
SEQUENCE PROPERTIES:	166

EXAMPLE 2 - STEAM TRAIN	167
SOUND SCRIPTS	168
TRAINZSCRIPT TUTORIAL	170

CHAPTER 12

Appendix A - Classes and Codes

CATEGORY CLASS	173

REGION CODES	176

Appendix B - Kinds and Containers

ACKNOWLEDGEMENTS	182

ADDITIONAL REFERENCE INFORMATION

Please read this document alongside the *Trainz Railroad Simulator 2006 User Manual* which forms part of the **TRS2006 installation**. This installed guide displays the config.txt and script source for all released TRS industry compatible assets. It also contains theory and tutorials.

The default location for the guide: C:\Program Files\Auran\TRS2004\Scripts\docs\index.chm Fix this paragraph references

The following additional information is available for download:

General assets

- Sample TRS in-game files including config files.
http://www.auran.com/trainz/creation/Trainz_custom.zip
- Sample 3D Studio Max and gmax files including textures.
http://www.auran.com/trainz/creation/source_files.zip

Rolling stock

- TRS asset PRR 40'Boxcar in-game files, source files and asset description.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Box_Car.zip
- TRS asset GATX-Oilco Tank Car in-game files, source files and asset description.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Tank_Car.zip
- TRS asset Coal Hopper in-game files, source files and asset description.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Coal_Hopper.zip
- TRS asset Container Flat in-game files, source files and asset description.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Container_Flat.zip

Interiors

- DD40x cabin interior in-game files, source files.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_dd40_interior.zip
- PB15 Steam interior in-game files, source files.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_PB15_interior.zip

Steam sound

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Steam_Sound.zip

Twinkles Particle Effects Editor

<http://www.auran.com/TRS2004/downloads/contentcreation/Twinkles.zip>

Other

- New TRS 3dsmax 4 & 5 Plugin for Bump mapping support and Bump mapping description.
http://www.auran.com/TRS2004/downloads/contentcreation/TRS_Max4_Plugin_Bump.zip
- Passenger Asset Tutorial.
http://files.auran.com/TRS2004/downloads/contentcreation/SP2-Passenger_Asset_Tutorial.zip
- Variable Rules Tutorial.
http://files.auran.com/TRS2004/downloads/contentcreation/SP2_VariableRulesTutorial.zip
- API Programmer's Reference Manual - covers most of the script classes and methods
<http://www.auran.com/TRS2004/trssp4dl/dpage.php>

CHAPTER 1

The Basics

INTRODUCTION

Welcome to the Content Creators Guide for Trainz Railroad Simulator 2006.

This document is designed to assist 3rd party content creators design and create functional content for Trainz Railroad Simulator 2006 (*TRS2006*).

The purpose of this document is to detail the way in which 3rd party content should be designed and built to be compatible with *TRS2006* and to describe the use of the new modules in TRS2006, the Content Manager Plus and the Content Creator Plus.

These Content Manager Plus module manages content on your computer, communicates with the Download Station and uploads or downloads content more easily than in previous versions of Trainz.

The Content Creator Plus module assists in creating model assets that are compatible with the Download Station requirements, and incorporates error checking that should assist in the creation of successful models. The module creates the config.txt file for the model asset, the entry of data is by means templates for each model asset Kind, and dialogue boxes filter and check the data.

This document provides the standards and procedures for a content creator to enter data for the model asset. It also provides advice on the creation of the model mesh using 3dsmax or gmax, and the creation of correct texture files for the model.

Scripting is an integral part of model creation, however details on script code and usage are included in a separate document. Brief references to Scripting are made in this document, as far as they effect the creation requirements.

Please check the Trainz Railroad Simulator website <http://www.auran.com/TRS2006/index.php> for additional information and updates. Major updates will be incorporated in future revisions of this document. A companion document "Content Creation Tips and Tricks" expands on some of the content covered here, and gives guidance to successfully export a working model into Trainz.

Please refer to the companion document "Content Creation Tips and Tricks 2006" for additional expanded information and guidance to improve your model creation.

OVERVIEW

What's new in TRS2006?

TRS2006 introduces a number of new functions specifically designed to enhance the performance of Trainz. The aim in this build is to make the train driving experience more realistic. Some of the new features in TRS2006 are.

In the Simulator:

- Cab Control Graphical Interface available when in Cab Control.
- Coupler breakage enhancements made to the physics engine, that require you to drive a train within the strength of couplers in the train. Should you exceed the strength capacity, couplers will break.
- Wheel slip is now included, you will need to allow for this in starting Trains.
- iTrainz in game chat allows users to talk to other users while playing.
- Added functionality on the Mini-map screen; query industry and consist information.
- New PaintShed with more flexible interface that is included as part of the installation.
- Bogey and bridge sounds supported.
- Re-styled Driver and Session menu screens allow for easier navigation of installed routes and sessions.

New Routes:

TRS2006 has many new routes provided by Auran staff and Third Party Creators. These introduce a new driving experience, with many new train and scenery models, and a number of driving scenarios for you to try.

For Managing Content:

The new Content Manager Plus module makes the management of assets very easy. It is integrated with the Download Station, listing the assets available on the Download Station, enabling updating of the list when connected to the Download Station via the Internet, and making the upload or download an efficient and simple process.

Content Manager Plus relies on a database of content in your installation on your computer. This database makes the loading of data into Trainz on start up very efficient, and reduces load time.

For Content Creation:

The Content Creator Plus module is integrated with the

Content manager Plus module. It is essential for creating content in TRS2006.

This document will touch on the use of Content Manager Plus but will cover in detail how to use Content Creator Plus to create config.txt files and manage the asset files.

The Basics of Content Creation

Creating new content for Trainz is a generally a five-step process.

1. Research

The research step involves finding out all the relevant information that you can about the item you wish to create. Research usually covers the accumulation of data about the content in question. The information needed may cover technical drawings of front, side and top, performance specifications (for loco engines), and photo's for texture reference.

You will generally find that much of this information can be obtained by searching on the Internet. You should also be aware of, and comply with, any copyright issues on information obtained, including pictures, textures and information.

2. Create a Mesh (.im file)

An .im file is an Indexed Mesh. These files are created by exporting from '3D Studio Max' (3dsmax) or 'gmax' using an Auran Jet Trainz plugin. 3dsmax requires a plugin to be separately installed, available for versions 4 and 5 from the Auran website:

<http://www.auran.com/TRS2006/index.php>

Gmax is a program created by Discreet as a game-specific version of their very popular '3D Studio Max' program. Gmax is available for free download from the [Discreet site](#). In order to use gmax with Trainz you will also need to download the Trainz 'Content Creation Pack' from the Auran website above. After installing and registering gmax, this pack installs into gmax and will enable you to export content directly into the .im file formats that Trainz uses.

Note: Previous versions of Trainz used a 'Progressive Mesh' file (.pm). Due to the new mesh reduction in TRS ('Level of Detail' mesh reduction) only .im files are used, progressive mesh files are obsolete. These files will still function in TRS2006, but for new models, the .im format is to be used.

So what is 3dsmax or gmax? These are 3D modeling programs that enable you to make things such as locomotives, items of rolling stock or scenery and trackside accessories. They are quite complex programs, and you can expect a steep learning curve should you decide to dive in and learn asset creation. However, on the plus side, the benefits are well worth it, and if you take the time to learn it well, you will certainly be able to

create some masterpieces.

The free gmax program lacks a few features of 3dsmax, notably a rendering option that allows you to see a rendered picture of the mesh asset in the program, while being developed, and the ability to export specular lighting and bump mapped textures. These exports enhance the visual effects of a locomotive boiler for instance, but nevertheless, gmax is able to do most things that a creator wishes, for models in Trainz.

3. Create textures. Creating textures for your assets is a very important part of the content creation process. Making good textures is one of the hardest things to do, but they can be the difference between a good-looking model and a great looking model.

Textures are created for Trainz using any 3rd party program that supports the creation of 2D art, like Adobe Photoshop or Paint Shop Pro.

There are a number of free programs that may be available, but they may lack the functionality of the commercial programs.

There are special requirements for texture types and sizes for models, details are given in this document.

4. Create a 'Config.txt' file (config.txt file). Each and every item of content for Trainz has it's own config.txt file. This file is a human-readable text file included with its corresponding item of content. Depending on what the item of content is will determine the necessary contents of the text file, but it will always contain a unique KUID, (a KUID is an identification serial code defined a little later in this document), a description and other information to make the model recognised by, and function in, Trainz.

Items of content created for Trainz are always assigned to a group of content called a KIND. A KIND is a type of content that has particular properties that Trainz recognizes. For example, one type of KIND is a TRACK. Trainz understands that items of content that belong to a group of this KIND define where locomotives and rolling stock travel, the "steel rails" for trains (of course some modern trains do not have "steel rails", the Maglev for instance. Other KINDs are listed further below in this document.

The config.txt file also includes a number of statements for content categories, time eras, Trainz Build, a list of items that the model may depend on (a KUID list of dependencies) and other instructions defining how the asset will behave in Trainz, or the Download Station how to manage the content.

These instructions are called "tags" and a number of "tags" may be grouped functionally into "containers" within the config.txt file. Refer to examples later in this document.

Previous versions of Trainz required the config.txt file to be generated by a text editor such as Notepad, and saved as UTF-8 code, not ANSI. This encoding option is available from the save dialogue box. Programs such as MS Word that can introduce unwanted formatting, including non standard quotation marks, were not used.

While it is possible to generate a config.txt file in this manner for TRS2006, the Content Creator Plus module will do this work for you when you enter the appropriate data.

This document gives details on how to create mesh and texture files, the config.txt file using the Content Creator Plus module, and how to incorporate them into the Trainz program.

5. Incorporate your asset into TRS2006

Previous versions of Trainz relied on a special directory (Downloads) where downloaded assets were stored. When Trainz initially started it added these files to a "cache". Subsequently, starting Trainz loaded this cache, which shortened the loading time considerably. Each time a new asset was installed, the cache was recalculated. If the Download Station Helper were used, the cach was automatically recalculated after every asset install.

Previous versions of Trainz also used a separate Custom model directory, to hold your newly created model files. This directory was not included in the cache, and each time Trainz started, files in this directory needed to be added to the cache of downloaded and installed models. If you had many models placed in this Custom directory, trainz could be very slow in loading.

TRS2006 does not use the installed Downloads directory nor the Custom directory for models. All model assets are encrypted and incorporated in Trainz as a total asset database. This makes for very fast load times, but can restrict easy access to your files.

Briefly, when a new asset is created, Content Creator Plus (CCP) creates a directory for the new asset under the "editing" directory in your installed Trainz, and saves a config file. This newly created directory can then be located, and files may be placed or exported for the asset mesh and textures. Even if the model is only partly constructed, it can be viewed in Trainz, by "committing" the asset into the database. The directory is then removed automatically, and Trainz can be launched to see the model.

On exiting Trainz, CCP can re-open the directory for work to continue on the model and files. This process will be fully explained in the following pages.

6. Upload your new content to the Auran Download Station.

You must create all models for TRS2006 using the CCP and CMP modules if you wish to upload to the Auran Download Station. Content Manager Plus has

an integrated function to package your model for the Download Station and upload it.

Content Manager Plus is a very useful application in that it automatically performs error checking and simplifies the preparation of your content for upload. The Content Manager Plus module embeds information into the upload package that is required by the Auran Download Station.

Don't include files such as .exe, .com or .bat in the model files. Because these files are a potential source of virus files they will not be packaged by CMP. Only in-game and text files are packaged.

For information on how to upload files and the Download Station requirements refer to Chapter 11.

7. Backup your new content.

Backing up newly created content is important, in case Trainz needs to be re-installed, or so that the content may be modified at a later date. Once content is committed into the database, the asset files are no longer available for copying or viewing, unless they are re-opened for editing.

1. Assets on the Download Station may be installed by re-downloading again. The Content Manager Plus (CMP) module makes this a seamless task.

2. To distribute model assets to other users, outside of the Download Station option, make a cdp file from within CMP.

3. Assets may be archived from within CMP. An archive name and location may be specified, and more than one asset may be incorporated in the archive. Trainz keeps track of any archive created, making an archive does not remove the assets from Trainz, it backs up the assets.

4. Of course the previous methods do not let you actually see the individual asset files nor allow them to be altered. To have access to the files, the asset is opened for editing in Explorer. An asset directory containing the files is created. This is found under the "editing" directory in the Trainz installation. If the asset has been given a username, the directory name will be that username. If a username has not been allocated, a directory name using alphabetic and numeric characters will be created.

The created directory can then be copied to another location on the computer for backup or other purpose. While creating an asset, this procedure may be useful each time before committing the asset, to ensure that no files are lost should a computer or program failure occur. Remember that texture files and exported mesh files are not easily accessible once the asset is committed.

5. A temporary directory could be created outside of the Trainz installation and the texture files and mesh files placed there. This directory can then be imported into CMP, the drag and drop feature makes this quite easy (drag the file directory into the main data window

of CMP). Of course a valid config.txt file needs to be included for CMP to recognise the directory as a valid asset. Opening the asset in CCP, correcting any errors, or adding other tags and containers to finish the model, and then saving the config.txt file will then create a build 2.5 asset.

WHAT IS A KUID?

A KUID is a unique number that is allocated to all content created for Trainz and can be thought of as much like a bar code.

KUID2 FORMAT

The KUID format in TRS2006 follows the standard adopted in TRS2004, and takes the form of three Identification Numbers (ID) each separated by a colon.

The breakdown of the new KUID system is as follows:

<KUID2:User_ID:Content_ID:Version_ID>

An example of a Kuid number with actual figures:

<KUID2:171456:38001:1>

User ID Number

The number 171456 after the KUID2 is the USER_ID of the content creator.

When you registered Trainz with the Planet Auran website, you would have been issued with your USER_ID. This is the number you should have entered into Trainz as your USER_ID.

Planet Auran may be found by LMB clicking on the Planet Auran symbol at the bottom of the Forum page, or visiting the website:

https://www.auran.com/planetauran/login_f.php

Every member of the Trainz community who is a member of Planet Auran has a USER_ID. Now, you may be wondering why you need a USER_ID if you don't intend to make any content for Trainz. If you intend to make a layout at some point in time and you'd like to share that layout with your friends or other community members, then you are in fact a content creator.

Content ID Number

The middle number is the CONTENT_ID.

This is a number that the content creator assigns to each creation to uniquely identify it, in previous Trainz builds. In TRS2006, the Content Creator Plus module will automatically assign a CONTENT_ID when you create a new asset. It will not repeat a number, and keeps track of all content numbers installed, so you do not have to keep a separate list or spread sheet of CONTENT_ID numbers for your model assets.

A CONTENT_ID number is also assigned automatically

when you save a layout (map). In these circumstances, number allocation will commence at 100,000 and increase for subsequent maps.

Note that the combination of a creator's USER_ID and the CONTENT_ID is unique, and will not conflict with assets created by others.

Content ID Ranges

Previous versions of Trainz used Content ID Ranges for KUID creation. The automatic assignment of CONTENT_ID numbers in TRS2006 has obsoleted the need for, or the usefulness of, a specific range of numbers for different kinds of assets.

Version ID Number

The third number is the asset Version_ID number. The default for all assets is 0 e.g.
<KUID2:xxx:yyyyy:0>

Should this asset require revisions after release to the Trainz Download Station, the Version ID for each subsequent revision may be updated as follows:

First revision	<KUID2:xxx:yyyyy:1>
Second revision	<KUID2:xxx:yyyyy:2>
Third revision	<KUID2:xxx:yyyyy:3> Etc.

The Maximum version number is 127.

After the maximum version number is used a new Content ID needs to be allocated for this asset, and the previous one needs to be added to the obsolete-table.

You may start a new asset at any version number, and you may miss numbers in a sequence, for example start at 1, the next one may be 3, then 4. Trainz will use the highest version number found for the asset. Obsoleting a KUID2 asset of the same content ID number does not require the use of the obsolete table.

Obsoleting is a process of replacing a previous item with a more recent one, for updating, improving, or replacing a faulty model.

Example 1:

A model <KUID2:171456:38001:3> is to be replaced by a newer version. The new number <KUID2:171456:38001:4> is used in the new config.txt file. The previous model will be replaced (obsoleted).

Example 2:

A model <KUID:171456:27001> (UTC version) is to be replaced, using the new KUID2 format, and with a new number, <KUID2:171456:27002:1>.

In this case, the obsolete table container is used to show the old model KUID to be obsoleted. In this way, the new KUID is linked to that of the older asset.

Important Notes:

1. <KUID2:xxx:yyyyy:0> is exactly equal to <KUID:xxx:yyyyy> in the old KUID format. These will be read as

duplicates should they be used simultaneously.

2. Similarly, <KUID2:xxx:yyyyy:1> acts as a KUID2 format obsoletion of <KUID:xxx:yyyyy> .

3. Using the zero, "0" as the first version is acceptable, however the display on the Download Station, and the installed file will be in the UTC KUID format, without the KUID2 format display. It is recommended that you start the numbering at one, "1" if this is a problem to you.

4. The Download Station displays a History of obsoleted models. If an obsoleted model has never been placed on the Download Station, do not include it in your obsolete table, it will result in a History error notification when you try to upload the model.

5. While you may use leading zeros in the KUID system, a version "02" will be the same as "002" or "2", and the zeros will be truncated. It is recommended not to use any leading zeros.

An asset placed in your map will display the latest installed version. When retrieving an asset from the Download Station, the newest version will be automatically provided, and the Download Station will display a History tag for the model versions. For further examples of the use of the obsolete-table, refer to [Page](#) .

TRAINZ BUILD

Trainz build is the numbering system allocated to each released version of Trainz. Content may be created for different versions, making use of the newer functions in a more recent release. Consequently, some models will not function in earlier versions of the simulator.

It is important for models to list the version of Trainz for which they are compatible. For older Trainz builds, this is done by entering a Trainz-Build number in the model config.txt file. The Content Creator Plus module will automatically add the correct trainz-build 2.5 to the config.txt file.

A model constructed for an older version may function in a newer version, but this is not guaranteed, as there have been amendments made to subsequent versions. Likewise, a model using the latest functions will not work in an earlier version.

The Download Station identifies which version of Trainz you have installed and when you use the Download Helper option, it will not allow the download of a more recent model that is incompatible with your version.

The approved Trainz-Build version numbers are;
trainz-build 1.3 for Trainz
trainz-build 1.5 for Ultimate Trainz Collection
trainz-build 2.4 for Trainz Railroad Simulator 2004
trainz-build 2.5 for Trainz Railroad Simulator 2006

CHAPTER 2

Introduction to Kinds, Containers, Tags, and Config.txt files

KINDS

Trainz recognises a number of Kinds of assets. Each Kind has different attributes, allowing different asset functions and behaviour to be recognised in the simulator.

In creating a new asset, a suitable Kind must be chosen. The following is a list of all the Kinds that may be created in TRS2006, with a brief description.

Activity: An activity is a scripted scenario, that details the locomotives and rollingstock used in a map, the driver settings, commands and scripts. A train driver can undertake a sequence of planned moves – a scenario.

Behavior: A configurable behavior module that forms part of a session.

Bogey: Bogeys are locomotive or rolling stock wheel mechanisms, sometimes known as 'Trucks'. This asset is for attachment to a traincar (locomotive or rollingstock) and can include animation and a shadow model.

Bridge: Road or rail bridges and similar assets, as variable length splines. The bridge kind may include initiator and terminator segments, and shadows. The height and gradient of the bridge spline may be varied in Surveyor.

Buildable: A variant of Kind Scenery, with similar attributes, but allowing attached track to be used as part of the model. It does not support processes, as used in a Kind Industry

Chunky-Track: Track and rails for Trains (the common flexi-track), by defining the cross section shape and properties of the track. It uses a texture file but does not require a 3dsmax or gmax mesh model.

Double-Track: Track splines that may place two or more tracks as one model, by specifying the track spacings to be used.

DriverCharacter: The locomotive driver character. This specifies the picture icon that appears in Driver for the engine driver.

DriverCommand: A command for the train driver to accomplish a specific task.

Engine: An engine specification for locomotives and rollingstock, which defines the detailed performance requirements; including throttle requirements, engine and braking performance

EngineSound: An engine sound specification, detailing the locomotive engine sound files, referenced by the enginesound tag in a traincar kind.

Environment: Additional sky textures, specifying the normal, night and stormy sky images to be used in Trainz

FixedTrack: A fixedtrack asset can be likened to a model trains sectional track system. The models may be straight or curved and snap into position when moved onto another track in Surveyor.

GroundTexture: A ground texture is tiled in Surveyor to color and cover the base grid. It can optionally reference a low polygon mesh and insert the mesh automatically as the ground is painted.

HornSound: A traincar horn sound, referenced by the hornsound tag in a traincar config file. It references the various sound file to be used.

Html-Asset: An html-asset example is the ingame tutorial. The config.txt file references one or more .html pages. The html-asset can be referenced from the scripts and from some of the Surveyor rules

Industry: A scenery asset with product processing functionality. Industry assets interact with compatible rolling stock assets through their script file and asset triggers. An Industry asset supports product queues and attached track

Interior: A traincar interior asset. It allows the interior mesh model to be defined, and has attached levers and controls to operate a locomotive in cab mode. It also creates an interior for rolling stock.

Library: Coded modules that interact with other coded modules.

Mesh: A mesh that is never referenced through Surveyor panels, but referenced from another asset. It could be referenced through the preview-mesh-kuid tag or as a kind attachment effect, like the red arrows used on fixed-track assets.

Mesh-Reducing-Track: Mesh-Reducing-Track is used to create poly efficient splines. The asset consists of a short high detailed mesh and a longer less detailed mesh, based on the same object. The short mesh is displayed when the camera is close to the asset whilst the long mesh is shown when less detail is required.

MOCrossing: Combined rail and road crossings, that reacts to trains or script control. This allows animation, special lighting effects and attachment points for rail track and roads.

MOJunction: Junction control levers, which are attached to track junctions, include sound, and may be offset a specified distance from the track. They can be used to replace the default junction lever.

MOSignal: A train signal with lights (coronas). It specifies the aspects the signal is capable of displaying, the light points activated when each state is displayed, and the corona details. The signal may be offset a specified distance from the track.

MOSpeedBoard: A Speed limit sign for Trains. It displays

the maximum limit (sign texture made by the creator) and the sign may be offset a specified distance from the track. The limit to control train speed is specified in the asset in metres per second.

Paintshed-Skin: A reskin texture for a locomotive or rolling stock asset.

Paintshed-Template: A template for particular locomotives and rolling stock that may be used in the integrated Paintshed utility. The template may be painted in different colour schemes

Pantograph: The animated mechanisms on the roof of electric locomotives that conduct electricity from the catenary (wires) above. It is referenced by the pantograph tag in a traincar config file.

Product: An individual product (commodity) that Trainz compatible rolling stock and industry assets are able to process. It specifies the type, unit of measurement and the picture icon that displays the product in the simulator. Produce and materials are product examples.

Product-Category: A category class of products (commodities) that Trainz compatible rolling-stock and industry assets are able to process. It specifies the type, unit of measurement and the picture icon that displays the category on Surveyor or Driver. Bulk, liquid, passengers and containers are product category examples.

Profile: A Profile is known as a Session in Trainz. This kind creates a session defining a single route with different consists, starting points, and industry outputs. Different sets of trains may be used in each different session.

Region: A region is chosen in Surveyor to create a new map or route. This Kind creates a new region in addition to the in-built regions, such as Australia or USA for example. The region can define geographical location, road traffic and weather conditions.

Scenery: A basic scenery asset, that supports night lighting, smoke (particle) effects, sound and animation. It is height adjustable and forms the majority of map objects used

Scenery-Trackside: A special scenery asset, attached to rail track, with the offset distance from the track specified in the asset. Examples could include a signal box, or dummy track sign or track object.

Steam-Engine: The special engine specification for steam locomotives, which defines the detailed performance requirements, including throttle settings, engine and braking performance, and boiler capacity and steam attributes.

Texture: A simple texture as an asset that can be referenced by from another asset for example, a custom corona, by reference to its kuid.

Texture-Group: Defines a group of textures as an asset that can be referenced from another asset or via scripting.

Track: Variable length spline based track, roads, and other scenery items. Tracks may include initiator and terminator segments, and are height adjustable. Other examples include fences, power lines and hedges.

TrackSound: A sound asset that is referenced by track or bogeys to play a different sound from the default track/train sound (for example, when a train travels over a bridge or through a tunnel).

Traincar: A locomotive or rolling stock item. A traincar specifies the dependant assets (bogey, engine sound, engine specification, pantograph and interior), to make a complete traincar asset.

Tunnel: Road and rail tunnel variable length splines. These allow the spline to be placed below ground and usually require an integrated initiator and termination mesh as a tunnel entrance.

TurnTable: A turntable asset for moving or rotating traincars, specifying the static and moving part of the turntable. Animated rotation (turntable) and lateral translation (transfer table) assets are supported.

Water2: Animated water texture assets.

There is one additional Kind that is used by Trainz, but may not be created from within Content Creator Plus (CCP), this is the Map Kind.

A map kind needs a number of files to be created when saving from Surveyor. This is not possible from within CCP. However, a Map Kind created in Surveyor may be edited in CCP, for instance, to add specific car Kuids to the Map file. These cars will then be mixed with the default in-built cars on roads in the map.

Config.txt Files

Each model asset that is created requires a config.txt file. This file is a simple text file that is used to describe the item of content to Trainz.

In previous Trainz Builds, the config.txt file was created in a text editor. TRS2006 now uses the Content Creator Plus module to create the required file, using a series of input boxes in a graphical user interface. This simplifies the creation of the file and allows the file to be checked for errors.

Because each different 'KIND' has a different config.txt requirement, please refer to the appropriate KIND descriptions and information on the following pages.

Containers

A data container is a portion of the config.txt file that covers a particular function for the model, for example the model mesh files to be used, or the effects to be applied.

The model asset has a top level container for data that is required for, or is common to, most model assets. Nested in this container may be other subcontainers, each describing a particular function of the asset.

Tags

Within a container the commands that Trainz recognizes are called Tags. Each tag indicates data values to be used or a function to be implemented.

Dialogue boxes and drop down menus are provided for the data entry and in-built error checking will indicate faulty data or entries. An error message display will assist in creating a correctly configured model.

Directory Structure

A new asset requires that the config.txt file, mesh files, texture files and other files be placed in a directory. This directory is created whenever a new asset is commenced.

The arrangement of files within the main directory or a subdirectory is, in most cases, the choice of the creator. It may be convenient to group some files in a subdirectory, for example, the night directory mesh and texture files. This can make it easy to distinguish the files associated with a particular part or function of an asset. The mesh table layout adopted since TRS2004, makes it easy to point the config.txt file data to the required subdirectories.

For some Kinds, certain files may be required to be placed in particular directories, and the creator has no choice in the layout of the directories. The creator will make the required subdirectories, and place the files. In these instances, requirements will be shown in the examples in the following Chapters and pages.

In other instances it may be more convenient to use only one main directory for all meshes and texture files for the asset. This can result in a smaller cdp file than if subdirectories were used. For example, sometimes a night mesh model may use the same texture files as the day mesh model. If these same textures are placed in the main directory and also in a night subdirectory, the files are actually loaded twice in the cdp package (the same file in different locations, or a different texture file using the same file name in a separate subdirectory, are converted to binary form in the packaging process, and are given a unique binary identifier).

Kinds and Container Relationship

Each Kind has unique attributes, but also shares some attributes with other Kinds. The tables below show the types of containers that are available in each Kind. These will indicate which asset Kind may be the most suitable

for a particular model to be created, and also show the hierarchy and relationship of Kinds and containers.

[Config File:](#) [Descriptions and usage](#)
[Original usage and Open in Explorer option](#)
[Autocreation in CCP](#)
[Sample config](#)

[Directory Structure:](#) Discussion on the directory structure for assets and the use of and need for subdirectories as required.

[Containers:](#) [Spreadsheet of Kinds and Containers showing hierarchy – Appendix.](#) Note there are approx 44 containers and 346 different tags in Trainz, a spreadsheet database of all tags, similar to the container listing below, is not possible in the document. However, each tag will be discussed within its appropriate container.

CONFIG.TXT FILES

Each item of content that you create is required to have a config.txt file. This file is a simple text file that is used to describe the item of content to Trainz. TRS assets have become far more flexible through the use of the new mesh-table fields.

Because each different 'KIND' has a different config.txt requirement, please refer to the appropriate KIND descriptions on the following pages.

Example Config.txt File, General breakdown:

General

While there is great flexibility in the order of placement of information in the config.txt file. It is important to ensure: the correct number and orientation of brackets; correct spaces within the statements are used; and the correct lower case or capitalisation is used for names.

While the tabbing of information across the page is not mandatory, it assists in the readability and debugging of the file. The Content Dispatcher will format your config.txt file in this manner. Do not remove any apparent blank line at the top of such a file if you edit it, as it contains hidden information used by Trainz.

kuid

Unique ID of this asset. The KUID contains basic creator information. Refer to page 3 for information on KUID format.

The New KUID2 format in TRS gives greater version control and flexibility over asset updates. It eliminates the need to give a new Content ID every time you the creator releases a new version of the same asset. Refer KUID2 information [Page 3](#).

mesh-table

This is the new and preferred method of asset mesh placement for most mesh asset types. It gives huge control over mesh placement and animations.

There are some asset types that **cannot use** a mesh-table. These include all Bridges, Tunnels, Rails, Pantographs and other Spline Objects (eg. Fences or Catenaries).

Important Note:

Any asset that uses a mesh-table will not be compatible with pre-TRS versions of Trainz (i.e. Ultimate Trainz Collection or UTC).
TRS will of course still read UTC assets.

Just remember that as with most major software releases, backwards compatibility is usually achievable, while forwards compatibility is often impossible.

default

Default is the 'main' mesh of the asset.

Blue text: Required tags Green text: Optional tags.

```
kuid <KUID2:1234:5678:1>
mesh-table
{
  default
  {
    mesh industry.lm
    anim anim.kin
    animation-loop-speed 1.0
    auto-create 1
    effects
    {
      0
      {
        kind name
        fontsize 0.15
        fontcolor 30,30,30
        att a.name0
        name name
      }
      1
      {
        kind corona
        att a.coronawhite
        frequency 1
        directional 0
        texture-kuid <KUID:-3:10111>
      }
    }
  }
  default-night
  {
    mesh nightwindows/nightwindows.im
    night-mesh-base default
  }
  attachedanimation
  {
    mesh meshname/meshname.im
    anim animname/animname.kin
    auto-create 1
    att a.pointname
    att-parent default
    animation-loop-speed 1.0
  }
}
kuid-table
{
  0 <KUID2:1234:6000:1>
  1 <KUID2:1234:6001:3>
}
obsolete-table
{
  0 <KUID:1234:5677>
}
preview-mesh-kuid <KUID:##:####>
username My Locomotive
description "You can have multiple lines
but no double quote characters in here.
Trainz automatically wraps this text. This
information is displayed with the model
on the Download Station. Please make the
description useful and informative."
region Australia
trainz-build 2.4
kind assetkind
category-class AD
category-region-0 AUS
category-region-1 UK
category-era-0 1960s
category-era-1 1970s
author Spock2204
organisation Trainz Thingz
contact-website http://www.auran.com/helpdesk
```

Note: This example refers to a .lm file (LOD file). The mesh tag could also refer to the mesh itself i.e.. industry.im. If this were the case the asset would not have LOD mesh reduction.

mesh

The 'main' mesh name. This may include a sub-path. ie: mesh nightwindows/nightwindows.im, where the file nightwindows.im has been placed in the subdirectory nightwindows.

Use .im files exported from 3dsmax or gmax (as opposed to .pm.) or reference an .lm file if you wish your asset to have 'Level of Detail' mesh reduction. A pantograph model still requires a .pm file.

Level of detail is a process of using different model meshes, depending on how far the viewer is from the model. A finer mesh is used for close up viewing in Trainz. For further information please refer to [Page 105](#).

anim

The animation file (.kin) exported from 3dsmax or gmax. This may include a sub-path.

animation-loop-speed 1.0

This tag must be here if the asset is to animate when placed. If this tag is not here when placed the animation will not play by default, but may play if controlled by script. A different value (e.g. 0.5, 2.0) may be used in the tag to play the animation at a different speed from that created in 3dsmax or gmax.

auto-create 1

The model is generated automatically when placed, or when you load a map which includes the model. In some instances you don't want the mesh visible (as this may be controlled through script). If auto-create is 0 the mesh will not be visible when placed.

default-night

'Main' night window mesh on scenery and industry and traincar assets. Modeled to the same 3d space as the default mesh and is inserted at the default mesh origin. Note that this example on the previous page has placed the mesh in a subdirectory.

night-mesh-base default

This night mesh is linked to the default mesh and is visible only at night. It is invisible if the 'default' mesh is invisible, (if the auto-create 0 line were used so the default mesh can be controlled by script).

att

The mesh (and animation if present) is inserted at a mesh attachment point rather than the origin (without this line the mesh is placed relative to the origin of the parent model).

att-parent name

The tag tells Trainz in which mesh the attachment point is located. The insertion attachment point is located within the mesh 'name', as listed in the config.txt.

kuid-table

A list of KUIDs required for this asset to function correctly.

A kuid-table must be included where the config.txt references additional KUIDs, such as a bogey, or

a pantograph, including Auran built-in KUIDs. The Download Station performs a search, and those found are added to the download pack.

obsolete-table

The obsolete-table describes the asset's revision history.

This field was used extensively for pre-TRS assets, as each version required a unique Content ID. However in order to make the content creators life a little easier, Trainz now uses the KUID2 format which adds another number as a version number. For KUID2 information see [Page 3](#).

TRS and the Trainz Download Station automatically detects the most recent version of an asset whether it be through the KUID2 Version ID or through the obsolete-table.

If there are no obsoletes, you can display the lines thus,

```
obsolete-table
{
}
```

or leave them out. While the blank entry may prompt you to add obsoletes when they are required, they are unnecessary lines in the config.txt file.

preview-mesh-kuid

Only add this to reference a different mesh for the Surveyor preview window. This is useful when an asset has a large bounding box. i.e. the Airport with it's jet animation. By using a different mesh it will fit better in the preview window. It can also reduce the polycount on screen.

username

The human-readable English name of this asset.

description

The human-readable multi-line English description of this asset. It displays on the Download Station with your model, so please make it useful and informative so others may understand your model, for instance, entering what the model is called and under which Region and category it may be found in Surveyor, is very useful in finding an installed model.

region

The country region to which this asset belongs. This should be one of the Auran-supplied region names

trainz-build

The Trainz build is the version number for which this asset was created. Refer to [Page 3](#) for further information.

kind

The asset kind. Must be one of the Auran-supplied asset kinds. i.e. kind industry

category-class

The class code for this asset. Refer to [Page 5](#).

category-region-0

A list of REGION codes or REGION GROUP codes,

starting at category-region-0 and progressing category-region-1, etc. with one code per tag. Do not miss out numbers in the sequence 0, 1, 2 etc.

category-era-0

A list of ERA codes, starting at category-era-0 and progressing category-era-1, etc. with one code per tag. Do not miss out numbers in the sequence 0, 1, 2 etc.

other entries

You may enter data for additional information at the bottom of the config.txt file, such as:

author, **contact-email** and **contact-website** are useful information, particularly if a user has a question on your models or would like to offer help or suggestions.

organisation name will show in Trainz in Railyard as the organisation for the model, for instance if you use Joe's Trainz or Cripple Creek Logging Company.

Note: Previous versions of Trainz config.txt also had an entry for creators licence details. This is not necessary in TRS2004, as your licence details are entered in the Content Dispatcher program, and these details will be transferred to your model config.txt file by the packaging process. Having a separate entry in the config.txt file will result in the licence details being entered twice in the config.txt file, in the installed asset.

EFFECTS (optional mesh-table variables)

EFFECT: KIND NAME

Some assets may have editable signs. When you set an asset's name in surveyor through the *Edit Properties* icon ('?' icon) the signage can be set-up to automatically update. The variables can be set for each sign.

Typical Kind Name mesh table set-up

```
mesh-table
{
  default
  {
    mesh industry.lm
    auto-create 1
    effects
    {
      0
      {
        kind name
        fontsize 0.15
        fontcolor 30,30,30
        att a.name0
        name name
      }
      1
      {
        kind name
        fontsize 0.3
        fontcolor 30,30,30
        att a.name1
        name name
      }
    }
  }
}
```

Note: Effects

You do not have to use the 0,1,2... block naming convention.

For the name effects described in the example, 0 could easily be *main*sign or something more relevant to your model.

Setting unique names may be useful for script purposes, so you can easily recognise a script reference for instance.

Breakdown of KIND NAME:

kind name

The effect kind

fontsize

The size of the sign text

fontcolor

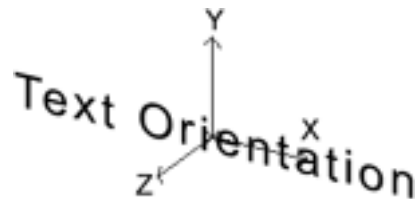
The colour of the sign text in r.g.b.

att

The sign text insertion point (part of the mesh). The attachment point must be orientated correctly in 3dsmax or gmax.

The diagram shows the correct orientation of the axis.

Points are placed in top view in 3dsmax/gmax. When aligning the point in 3dsmax or gmax, click on the Hierarchy - Affect Pivot Only option to examine the axis direction, turn off the Affect Pivot Only selection, and rotate the point itself to the correct orientation. Click on the Affect Pivot Only option again to check the new alignment. Make sure you rotate the point, not the axis.



name

The default text when placed

When **name** name is specified, it uses the asset's changeable name. Changed through the *Edit Properties* icon (the '?' icon) in Surveyor.

If **name Graceland** (for example) was used, the sign would never be able to be changed even though the user may have changed the asset's name in Surveyor.

When you use the name tag, this name will appear in the minimap. In some instances the names will be hard to distinguish in a crowded map. For this reason, it may be advisable to limit the use of the name tag to objects that have a use for the attribute.

EFFECT: KIND CORONA

A corona is a 'glow' light effect. It is a simple texture that is inserted at an attachment point within the mesh. Coronas can be added to any asset that uses a mesh-table.

Examples of coronas used in-game can be seen on the Airport and Airport basic assets. The Jumbo Jet, the Cessna and the Airport tower all use flashing coronas.

Breakdown of KIND CORONA:

kind corona (required)

The effect kind

att (required)

The corona insertion point in the main mesh.

Typical Kind Corona mesh-table set-up

```
mesh-table
{
  default
  {
    mesh asset.lm
    auto-create 1
    effects
    {
      0
      {
        kind corona
        att a.corona0
        texture-kuid <KUID:-3:10110>
        frequency 2
        directional 0
        object-size 0.20
      }
      1
      {
        kind corona
        att a.corona1
      }
    }
  }
}
```

texture-kuid (optional)

Add this tag only when you want to specify your own texture for the corona. It specifies the KUID of a kind texture asset. See [KIND TEXTURE](#).

If the texture-kuid tag is not present the corona will use the default yellow/orange texture in TRS.

frequency 2 (optional)

This variable specifies the frequency in Hz (or 'flashes' per second). e.g. 1 for once per second, 0.5 for once every 2 seconds, 2 for twice per second.

directional 0 (optional)

The default for coronas is to be aligned to the attachment point to face the NEGATIVE Z direction. This is especially useful for Traincars.

Adding this tag over-rides this behavior, to make the corona always face the screen – useful for scenery objects.

object-size 0.15 (optional)

Size of the corona on the object when viewed up close

Defaults to 0.15 (i.e. 0.15m)

TRS released corona textures:

- Yellow/orange corona Default (if no texture-kuid)
- Green corona <KUID:-3:10110>
- Red corona <KUID:-3:10112>
- White corona <KUID:-3:10111>

EFFECT: KIND TEXTURE-REPLACEMENT

This effect was created for rolling stock items to swap the visible texture of bulk loads (such as coal or woodchips).

If a coal car is set up to take any bulk load (which includes woodchips) the 'coal' texture on the load mesh will update to a 'woodchips' texture when it loads woodchips.

For detailed information on the setup of rolling stock load, see [Page 107](#).

EFFECT: KIND ATTACHMENT

In TRS we now have the ability to attach a mesh into another mesh by referencing it's KUID through a mesh-table. Prime examples are the TRS fixed-track assets.

Rather than having an arrow in each fixed-track asset directory, we saved a lot of memory space by making the config.txt file reference a KUID of that mesh. Therefore, it only needed to be cached once. The mesh to be used should only be of kind scenery or kind mesh.

Kind Attachment example (fixed-track)

```
username FT 10 Deg 700m Rad
kind fixedtrack
kuid <KUID2:####:#####:1>
region Fixed track
type Fixed track
preview-mesh-kuid <KUID:-3:10099>

mesh-table
{
  default
  {
    mesh 10-700.im
    auto-create 1
    effects
    {
      arrow0
      {
        kind attachment
        att a.track0a
        default-mesh <KUID:-3:10092>
        surveyor-only 1
      }
      arrow1
      {
        kind attachment
        att a.track0e
        default-mesh <KUID:-3:10092>
        surveyor-only 1
      }
    }
  }
}

attached-track
{
  track0
  {
    track <KUID:-1:15>
    vertices
    {
      0 a.track0a
      1 a.track0c
      2 a.track0e
    }
  }
}
```

WARNING: Never cross-reference a kind attachment KUID with the assets own KUID. Unless of course you want to show your friends what an instant fatal error looks like!

Fixed Track configuration used in the example

Arrow mesh config.txt (for reference)

```
kuid <KUID:-3:10092>
kind mesh
mesh-table
{
  default
  {
    mesh arrow.im
    auto-create 1
  }
}
```



Breakdown of KIND ATTACHMENT:

Note that the example covers the information relating to the attachment. Additional entries for category region, era, trainz-build, etc. are necessary to make a complete config.txt file.

type

Specify a type for the model that will be used in the Surveyor menu drop down menu for Track or Object type.

kind attachment (required)

The effect kind

att (required)

The insertion point of the attached mesh. In the example left, it is the origin of the 'default' mesh.

default-mesh <KUID2:####:#####:1> (required)

The KUID of the attached mesh.

surveyor-only 1 (optional)

Adding this means the attached mesh will only be visible in Surveyor and not Driver.

For tracks, this is especially handy so you know when it starts and ends, but it won't be there when driving around.

attached-track

Information on the track to be attached to the model by Trainz. This includes a name for the track (track0), the track KUID to be used, and attachment points placed in the 3dsmax/gmax model to define the curve for the attached track to follow.

The points use the a.name naming convention, any names may be used, but using the "track" name (a.track0a) for rails, and the "road" name (a.road0a) is convenient. A specified road spline connection will generate road traffic.

See [Page 82](#) for more details on using the attachment points and the special orientation of the axis at the ends of the track.

EFFECT: KIND ANIMATION

This effect is used when a mesh has a variety of animations. Usually the animations will be controlled by a script related to the asset.

An example of the kind animation effect is the PB15 interior coalman. The script for this ties in the animations with the coal requirements of the steam locomotive.

```
coalman
{
  mesh coalman/coalman.im
  auto-create 1
  att-parent default
  att a.coalman
  effects
  {
    shovel
    {
      kind animation
      anim coalman/coalman_shovel.kin
    }
    wave
    {
      kind animation
      anim coalman/Coalman_wave.kin
    }
  }
  Etc...
```

Breakdown of KIND ANIMATION:

kind animation (required)

The effect kind

anim (required)

Reference to the animation file (.kin)

looped 1 (optional)

Use only if the animation is looping (repeating).

Default 0 (i.e. not looped).

speed 1 (optional)

Speed factor of the animation.

Default 1

2 = Double speed

0.5 = Half speed

The PB15 interior script can be viewed on:

[Page 59](#),

The PB15 config.txt can be viewed on:

[Page 55](#)

KIND: HTML-ASSET

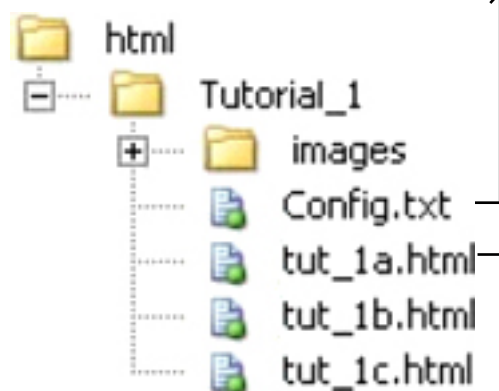
A html-asset can be referenced from the scripts and from some of the Surveyor *rules* (i.e. you select the html-asset by name, then type in the name of the *.html file in the small edit box).

A typical example of KIND: html-asset in use is the in-game tutorial.

The general setup is very simple, you bundle your config.txt along with one or more .html pages.

Config.txt

```
kind html-asset
username "my html page"
kuid <KUID2:####:#####:0>
```



tut_1a.html

```
<HTML>
  <BODY>
    <IMG SRC='images/tut_1a.jpg'>
  <BODY>
</HTML>
```

KIND: PRODUCT

KIND: PRODUCT-CATEGORY

OVERVIEW

Products are the commodities that TRS compatible rolling stock and industry assets process. TRS released products fall under one of four product-categories... Liquid Load, Bulk load, Container or Passenger.

This document description is not only to give you the list of products and product categories, but to also give you a better understanding of how the products are used.

As product-categories are fundamentally different to products, so too are they dealt with differently. A rolling stock item may be able to pick up anything that falls under a product-category or be limited to one or a few products only.

For example, passenger cars can only take passengers, where-as a gondola may be set up to take any bulk load available.

IN-GAME VISUALISATION OF PRODUCTS.

In TRS, products can be displayed in a few ways:

1) An animated load representation.

This technique is used for bulk-category loads such as coal or woodchip products both in industry and rolling stock assets and for liquid loads through indicators adjacent to storage tanks. The animation is non-looping. Say we have an industry bulk load animation with the frames running from 0 to 30. Empty will be at frame 0 and full will be at frame 30.

Texture swapping is possible for some rolling stock bulk loading assets. Details of how the texture swapping is set up is available on [Page 107](#).

2) Mesh attachment representation.

This technique is used for container-category loads such. 20ft and 40ft Containers, General Goods, Lumber and Logs all use this technique. If a piece of rolling stock has the potential to carry several product types (such as a flat car), it is possible to set up the loads to be mutually exclusive through it's config. That is, if it has capacity of one load, it cannot load any other product types.

3) 'View details' Driver information window display.

This (of course) can be used for all rolling stock items, but specifically, it is the means to see the load of rolling stock that cannot otherwise visually display it's load. i.e.. Tank Cars and enclosed Box Cars.

Box cars can be setup to take General Goods but are constructed *without* load attachments.

Note: Tank cars and tenders may use a separate animated 'loader' mesh to visualise the loading of liquids. This is set up through the industry asset's script and the rolling stock item's config. For script reference please refer to *index.chm* found in `\Trainz\scripts\docs` directory.

Refer: KIND TRAINCAR for links to in-game examples, descriptions and source files of the various types of product compatible rolling stock.

TRS RELEASED PRODUCT-CATEGORY LIST (kind product-category)

- **Container** <KUID:-3:10042>
- **Bulk Load** <KUID:-3:10040>
- **Liquid Load** <KUID:-3:10044>
- **Passenger** <KUID:-3:10091>

Container Category Config.txt

```
kind product-category  
kuid <KUID:-3:10042>  
username Container"
```

Bulk Load Category Config.txt

```
kind product-category  
kuid <KUID:-3:10040>  
username "Bulk Load"
```

Passenger Category Config.txt

```
kind product-category  
kuid <KUID:-3:10091>  
username "Passenger"
```

Liquid Load Category Config.txt

```
kind product-category  
kuid <KUID:-3:10044>  
username "Liquid Load"
```

TRS RELEASED PRODUCT LIST (kind product)

Product Category Container:

- 20ft Container <KUID:-3:10014>
- 40ft Container <KUID:-3:10041>
- General Goods <KUID:-3:10013>
- Log <KUID:-3:10001>
- Lumber <KUID:-3:10003>

Product Category Bulk Load:

- **Coal** <KUID:44179:60013>
- **Woodchips** <KUID:-3:10002>

Product Category Liquid Load:

- Aviation Fuel <KUID:-3:10045>
- Crude Oil <KUID:-3:10010>
- Diesel Fuel <KUID:-3:10011>
- Petrol Fuel <KUID:-3:10012>
- Water <KUID:-3:10004>

Product Category Passenger:

- Passenger <KUID:-3:10060>

PRODUCT CONFIG.TXT FILES

Legend:

kind product = New kind in TRS

product-category = KUID of applicable category for this product

instance-type resource = Used when there is no mesh (or only one mesh) is referenced in the mesh table (i.e. Liquids, Bulk loads etc.).

instance-type instance = Used when more than one mesh is in the mesh table i.e: Passengers, General Goods. *200 max. 'size' per Asset.*

icon-texture = the in-game representation of the product when specifying the load type for a compatible rolling stock item (in Driver).

mass = The physical mass of the product (or is that 'virtual' mass?) ☺

- For Containers and Passengers this is calculated in *kilograms/unit*
- For Liquid and Bulk loads this is calculated in *kilograms/litre*

product-texture = The texture to be used with load 'texture-replacement'. i.e. When a hopper loads woodchips instead of it's default load of coal. Refer to *Load_Texture_Replacement.pdf* for details.

allows-mixing 1 = Products with this tag may be combined in a single queue along with other products of the same category. Eg. Lumber and 20ft Container on a flatcar. By default, allows-mixing is set to 0. Therefore by default, a queue will only allow one product-category at a time.

- To look at allows-mixing from another angle, liquid products should never have allows-mixing enabled. Otherwise you have the potential to mix petrol with oil all within the same tanker, and I don't think cars like 2-stroke fuel too much! ☺

We have placed Products and Product-Category files in the 'scenarios' directory. (*This location is not mandatory though*).

AVIATION FUEL PRODUCT

Aviation Fuel Product Config.txt

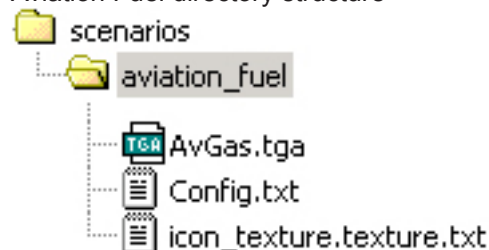
```
kind product
kuid <KUID:-3:10045>
username "Aviation Fuel"

product-category <KUID:-3:10044>
instance-type resource
icon-texture "icon_texture.texture"

mass 0.800

mesh-table
{
}
```

Aviation Fuel directory structure



icon_texture.texture.txt

```
Primary=AvGas.tga
Alpha=AvGas.tga
Tile=none
```



AvGas.tga
64x64 32 bit

COAL PRODUCT

Coal Product Config.txt

```
kind product
kuid <KUID:44179:60013>
username "Coal"

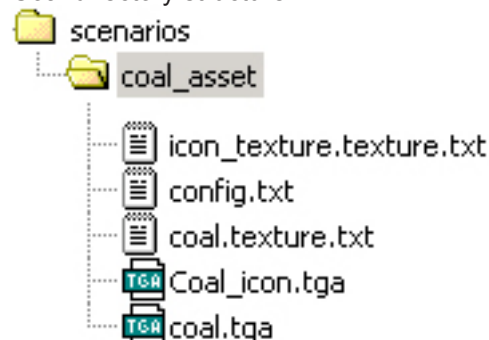
instance-type resource
product-category <KUID:-3:10040>
icon-texture "icon_texture.texture"

mass 0.860

product-texture "coal.texture"

mesh-table
{
}
```

Coal directory structure



icon_texture.texture.txt

```
Primary=Coal_icon.tga
Alpha=Coal_icon.tga
Tile=none
```



Coal_icon.tga
64x64 32 bit

GENERAL GOODS PRODUCT

General Goods Product Config.txt

```
kind product
kuid <KUID:-3:10013>
username "General Goods"

product-category <KUID:-3:10042>
instance-type instance
icon-texture "icon_texture.texture"

mass 1400

mesh-table
{
  default
  {
    mesh general_goods.im
  }
  crate2
  {
    mesh crate2.im
  }
  crate3
  {
    mesh crate3.im
  }
  crate4
  {
    mesh crate4.im
  }
  crate5
  {
    mesh crate5.im
  }
  crate6
  {
    mesh crate6.im
  }
  crate7
  {
  }
}

allows-mixing 1
```

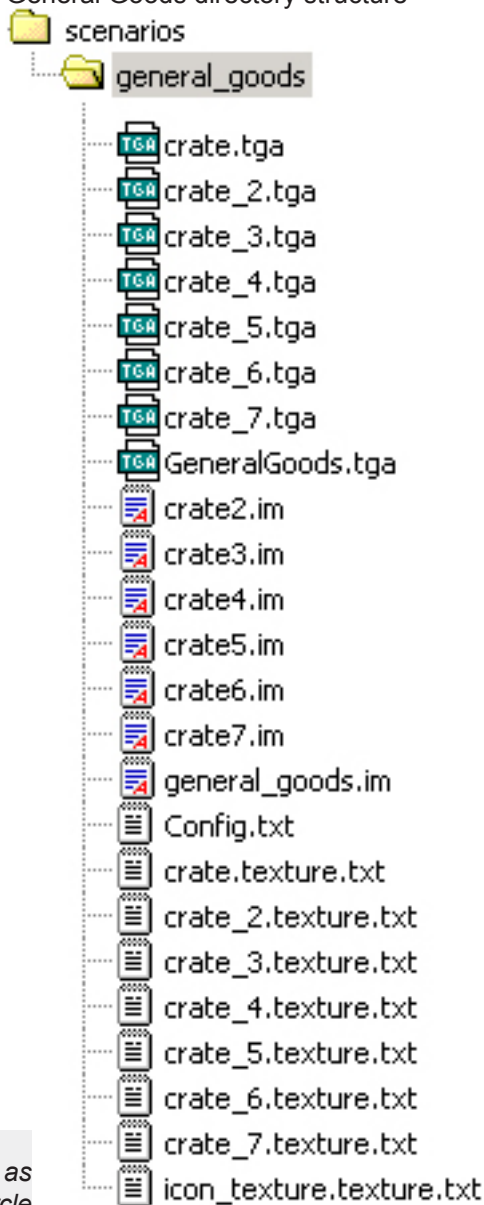
icon_texture.texture.txt

```
Primary=GeneralGoods.tga
Alpha=GeneralGoods.tga
Tile=none
```



GeneralGoods.tga
64x64 32 bit

General Goods directory structure



Note: The circular icon is made using an opacity map as the Alpha channel in the .tga file. This is a white circle and black background to cut out the circular shape. Both Primary and Alpha lines in the texture.txt file refer to the same .tga file.

General Goods Mesh Dimensions

Length	1.6 metres
Width	1.6 metres
Height	2.8 metres

CRUDE OIL PRODUCT

Crude Oil Product Config.txt

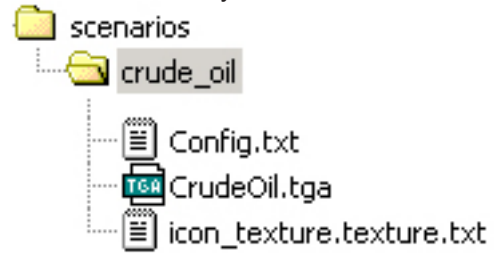
```
kind product
kuid <KUID:-3:10010>
username "Crude Oil"

product-category <KUID:-3:10044>
instance-type resource
icon-texture "icon_texture.texture"

mass 0.9

mesh-table
{
}
```

Crude Oil directory structure



icon_texture.texture.txt

```
Primary=CrudeOil.tga
Alpha=CrudeOil.tga
Tile=none
```



CrudeOil.tga
64x64 32 bit

DIESEL FUEL PRODUCT

Diesel Fuel Product Config.txt

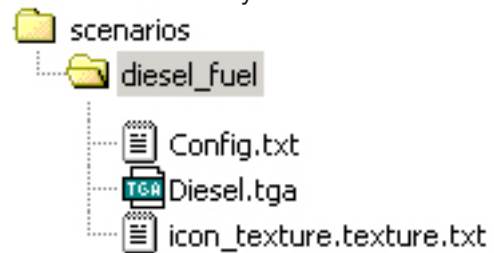
```
kind product
kuid <KUID:-3:10011>
username "Diesel Fuel"

product-category <KUID:-3:10044>
instance-type resource
icon-texture "icon_texture.texture"

mass 0.830

mesh-table
{
}
```

Diesel Fuel directory structure



icon_texture.texture.txt

```
Primary=Diesel.tga
Alpha=Diesel.tga
Tile=none
```



Diesel.tga
64x64 32 bit

20 FT CONTAINER PRODUCT

20ft Container Product Config.txt

```
kind product
kuid <KUID:-3:10014>
username "20ft Container"

product-category <KUID:-3:10042>
instance-type instance
icon-texture "icon_texture.texture"

mass 11000

mesh-table
{
  default
  {
    mesh 20ft_container.im
  }
  20ft_Pil
  {
    mesh 20ft_Pil.im
  }
  20ft_Capital
  {
    mesh 20ft_Capital.im
  }
  20ft_matsui
  {
    mesh 20ft_matsui.im
  }
  20ft_Gen1
  {
    mesh 20ft_Gen1.im
  }
}

allows-mixing 1
```

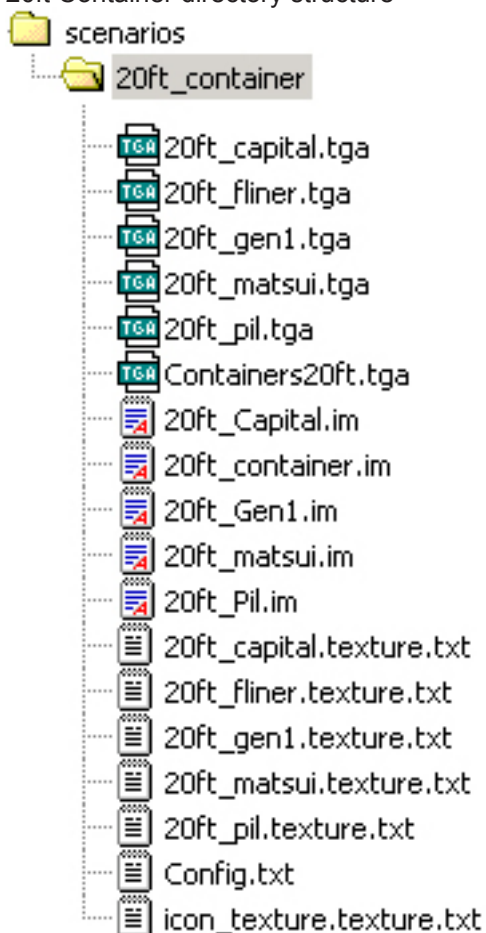
icon_texture.texture.txt

```
Primary=Containers20ft.tga
Alpha=Containers20ft.tga
Tile=none
```



Containers20ft.tga
64x64 32 bit

20ft Container directory structure



20 ft CONTAINER Mesh Dimensions

Length	6.10 metres
Width	2.44 metres
Height	3.05 metres

40 FT CONTAINER PRODUCT

40ft Container Product Config.txt

```
kind product
kuid <KUID:-3:10041>
username "40ft Container"

product-category <KUID:-3:10042>
instance-type instance
icon-texture "icon_texture.texture"

mass 22000

mesh-table
{
  default
  {
    mesh 40ft_container.im
  }
  pils
  {
    mesh 40ft_pils.im
  }
  matsui
  {
    mesh 40ft_matsui.im
  }
  capital
  {
  }
  blue
  {
    mesh 40ft_blue.im
  }
}

allows-mixing 1
```

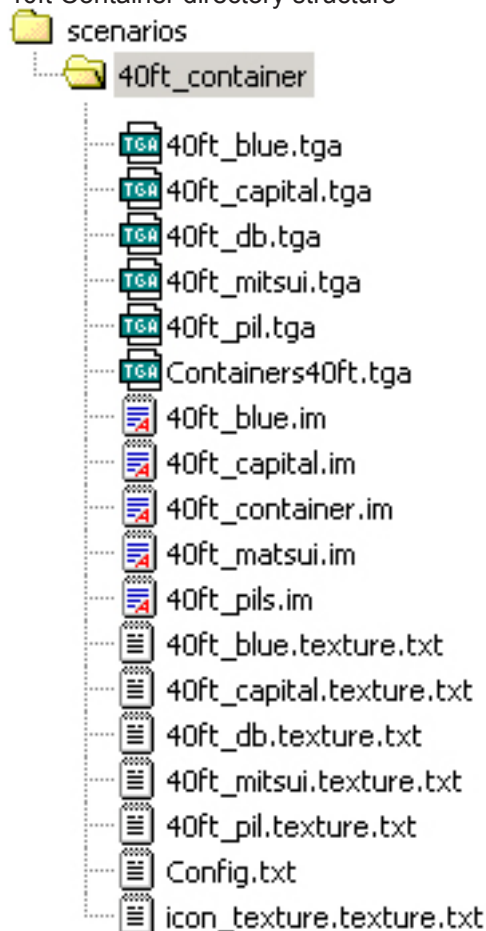
icon_texture.texture.txt

```
Primary=Containers40ft.tga
Alpha=Containers40ft.tga
Tile=none
```



Containers40ft.tga
64x64 32 bit

40ft Container directory structure



40 ft CONTAINER Mesh Dimensions

Length	12.20 metres
Width	2.44 metres
Height	3.05 metres

LUMBER PRODUCT

Lumber Product Config.txt

```
kind product
kuid <KUID:-3:10003>
username "Lumber"
product-category <KUID:-3:10042>

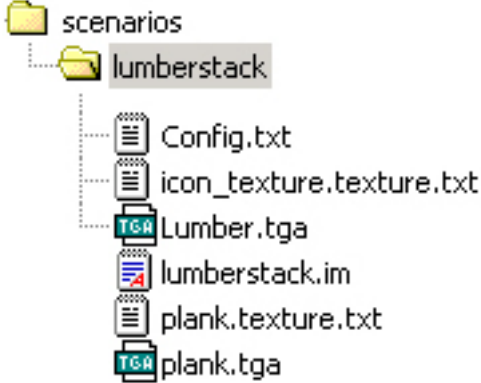
instance-type resource
icon-texture "icon_texture.texture"

mass 8000

mesh-table
{
  default
  {
    mesh lumberstack.im
  }
}

allows-mixing 1
```

Lumber directory structure



icon_texture.texture.txt

```
Primary=Lumber.tga
Alpha=Lumber.tga
Tile=none
```



Lumber.tga
64x64 32 bit

Log Mesh Dimensions

Length	6.0 metres
Diameter	1.2 metres

Lumber Mesh Dimensions

Length	6.10 metres
Width	2.13 metres
Height	1.62 metres

PETROL FUEL PRODUCT

Petrol Fuel Product Config.txt

```
kind product
kuid <KUID:-3:10012>
username "Petrol Fuel"

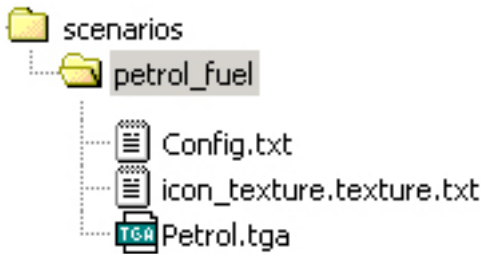
product-category <KUID:-3:10044>
instance-type resource
icon-texture "icon_texture.texture"

mass 0.7

mesh-table
{
}


```

Petrol directory structure



icon_texture.texture.txt

```
Primary=Petrol.tga
Alpha=Petrol.tga
Tile=none
```



Petrol.tga
64x64 32 bit

WATER PRODUCT

Water Product Config.txt

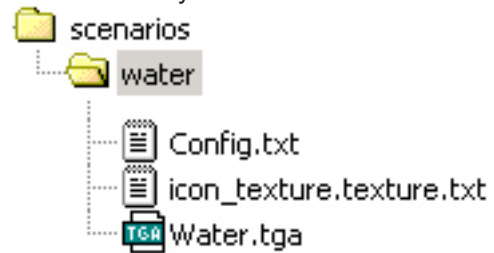
```
kind product
kuid <KUID:-3:10004>
username "Water"

product-category <KUID:-3:10044>
instance-type resource
icon-texture "icon_texture.texture"

mass 1.0

mesh-table
{
}
```

Water directory structure



icon_texture.texture.txt

```
Primary=Water.tga
Alpha=Water.tga
Tile=none
```



Water.tga
64x64 32 bit

WOODCHIPS PRODUCT

Woodchips Product Config.txt

```
kind product
kuid <KUID:-3:10002>
username "Woodchips"

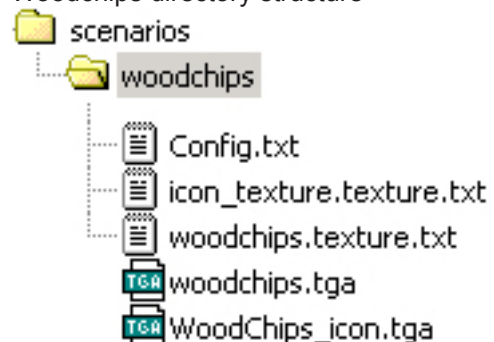
instance-type resource
product-category <KUID:-3:10040>
icon-texture "icon_texture.texture"

mass 0.400

product-texture "woodchips.texture"

mesh-table
{
}
```

Woodchips directory structure



icon_texture.texture.txt

```
Primary=WoodChips_icon.tga
Alpha=WoodChips_icon.tga
Tile=none
```



WoodChips_icon.tga
64x64 32 bit

PASSENGER PRODUCT

Passenger Product is unique as it is controlled very differently from other products. All passenger animations, meshes and texture files are located within the passenger directory. There was simply too many to list in this doc. ☺

icon_texture.texture.txt

```
Primary=passengers.tga
Alpha=passengers.tga
Tile=none
```



passengers.tga
64x64 32 bit

Passenger Product Config.txt - Extract

```
kind product
kuid <KUID:-3:10060>
name "Passenger"
product-category <KUID:-3:10091>
instance-type instance
icon-texture "icon_texture.texture"

mass 65

mesh-table
{
  male-suit
  {
    mesh MaleSuit.im
    effects
    {
      walk
      {
        kind animation
        anim MaleSuit_walk.kin
        looped 1
      }
    }
    pissedoff
    {
      kind animation
      anim MaleSuit_pissedoff.kin
    }
    lookatwatch
    {
      kind animation
      anim MaleSuit_lookatwatch.kin
    }
    lookdownline
    {
      kind animation
      anim MaleSuit_lookdownline.kin
    }
    shuffle
    {
      kind animation
      anim MaleSuit_shuffle.kin
    }
    sitdown
    {
      kind animation
      anim MaleSuit_sitdown.kin
    }
    sitdownloop1
    {
      kind animation
      anim MaleSuit_sitdownloop1.kin
    }
    sitdownloop2
    {
      kind animation
      anim MaleSuit_sitdownloop2.kin
    }
  }
  FemaleLongDress
  {
    mesh FemaleLongDress.im
    effects
    {
      walk
      {
        kind animation
        anim FemaleLongDress_walk.kin
        looped 1
      }
    }
    lookatwatch
    {
      kind animation
      anim FemaleLongDress_lookatwatch.kin
      looped 1
    }
  }
  FemaleLongDressFat
  FemaleLongDressBaby
  Female_Pants
  FemaleShortDress
  FemaleSuit
  KidFemale
  KidMale
  MaleFat
  MaleShirt
  MaleShorts
}
```

KIND: ENGINE

An engine file specifies a traincar's performance and physics parameters. TRS now supports steam physics and thus, steam engine files have additional fields for this. The following examples are of the SD40-2 Diesel locomotive and the PB-15 Steam locomotive.

ENGINE FILE (SD40-2 Diesel locomotive)

Default location: C:\Program Files\Auran\TRS2004\Engines

WARNING: ALTERING ENGINE FIGURES MAY RESULT IN UNDESIRE EFFECTS IN PERFORMNACE AND BEHAVIOR OF YOUR TRAINS. (MAKE BACK-UP COPIES OF YOUR ENGINE CONFIG FILES!!)

```
kuid <KUID:-1:42004221>
kind engine
rem SD402
flowsize
{
    trainbrakepipe 170000
    epreservoirpipe 0.1
    no3pipe 0.1
    no4pipe 0.1
    auxreservoirvent 0.1
    auxreservoir_no3 0.1
    auxreservoir_trainbrakepipe 0.1
    autobrakecylindervent 0.1
    auxreservoir_autobrakecylinder 0.1
    equaliser_mainreservoir 0.06
    equaliservent 0.06
    equaliserventhandleoff 0.1
    equaliserventemergency 0.1
    no3pipevent 1.5
    no3pipe_mainreservoir 0.1
    compressor 10
    trainbrakepipe_reservoir 1
    trainbrakepipevent 0.06
    no3pipe_autobrakecylinder 0.1
    epreservoirpipe_autobrakecylinder 0.1
    mainreservoir_ep 0.1
    vacuumbrakepipe 0.1
    vacuumbrakepipereleasevent 0.1
    vacuumbrakepipevent 0.1
    vacuumbrakereservoir_vacuumbrakepipe 0.1
    vacuumbrakecylinder_vacuumbrakepipe 0.1
    highspeedexhauster_vacuumbrakepipe 0.1
}
volume
{
    scale 1
    trainbrakepipe 0.2
    epreservoirpipe 0.2
    no3pipe 0.2
    no4pipe 0.2
    auxreservoir 0.0384678
    autobrakecylinder 0.00969387
    vacuumbrakepipe 0
    vacuumbrakereservoir 0
    vacuumbrakecylinder 0
    mainreservoir 0.9
    equaliser 0.5
    independantbrakecylinder 0.0103239
}
pressure
{
    scale 1
    compressor 0.00946941
    mainreservoir 0.00946941
    highspeedexhauster 0
    brakepipe 0.00736041
    brakeinitial 0.00693861
    brakefull 0.0044992
    indbrakefull 0.005075
```

```
    trainbrakepipe_start 0.00553261
    epreservoirpipe_start 0
    no3pipe_start 0
    no4pipe_start 0
    auxreservoir_start 0.00553261
    autobrakecylinder_start 0.00560291
    vacuumbrakepipe_start 0
    vacuumbrakereservoir_start 0
    vacuumbrakecylinder_start 0
    mainreservoir_start 0.00946941
    equaliser_start 0.00553261
    independantbrakecylinder_start 0.00560291
}
mass
{
    scale 1
    fuel 6.2156e+006
}
motor
{
    resistance 1.7
    adhesion 7
    maxvoltage 600
    maxspeed 40
    brakeratio 55000
    max-accel 3500
    max-decel 9000
    axle-count 6
    surface-area 80
    moving-friction-coefficient .03
    air-drag-coefficient .00017
}

throttle-power
{
    0 {
        0 0
    }
    1 {
        0 107
        2.2 62
        4.4 34
        6.6 31
        8.8 25
        13.3 18
        22.2 11
    }
    2 {
        0 224
        2.2 125
        4.4 68
        6.6 62
        8.8 50
        13.3 36
        22.2 22
    }
}
```

Continues next page...

```

3 {
    0      373
    2.2    187
    4.4    125
    6.6     93
    8.8     75
    13.3    53
}

4 {
    0      448
    2.2    249
    4.4    166
    6.6    125
    8.8    100
    13.3    71
    22.2    45
}

5 {
    0      618
    2.2    309
    4.4    206
    6.6    154
    8.8    123
    13.3    88
    22.2    56
    35.5    36
}

6 {
    0      747
    2.2    374
    4.4    249
    6.6    187
    8.8    149
    13.3    107
    22.2    68
    35.5    44
}

7 {
    0      872
    2.2    436
    4.4    291
    6.6    218
    8.8    174
    13.3    124
    22.2    79
    35.5    51
    44.4    42
}

8 {
    0      996
    2.2    498
    4.4    332
    6.6    249
    8.8    199
    13.3    142
    22.2    90
    35.5    58
    44.4    48
}

}

dynamic-brake

```

```

{
    0 {
        0      0
    }

    1 {
        1.333  0
        2      30
        5      25
        10     15
        12     0
    }

    2 {
        1.333  0
        3      50
        10     35
        14     20
        15     0
    }

    3 {
        1.333  0
        3      60
        10     40
        17     20
        22     0
    }

    4 {
        1.333  0
        4      80
        10     60
        20     20
        25     0
    }

    5 {
        1.333  0
        5      190
        10     70
        25     25
        29     0
    }

    6 {
        1.333  0
        5      250
        10     80
        29     70
        32     0
    }

    7 {
        1.333  0
        5      250
        10     100
        32     60
        36     0
    }

    8 {
        1.33  0
        5      250
        10     100
        36     50
        40     0
    }

}

```

dynamic-brake

ENGINE FILE (PB15 locomotive)

Default location - C:\Program Files\Auran\TRS2004\Engines

```
kuid <KUID:44179:51002>
kind steam-engine
rem PB15
epbrakes 1
flowsize
{
  trainbrakepipe 170000
  epreservoirpipe 0.1
  no3pipe 0.1
  no4pipe 0.1
  auxreservoirvent 0.1
  auxreservoir_no3 0.1
  auxreservoir_trainbrakepipe 0.1
  autobrakecylindervent 0.1
  auxreservoir_autobrakecylinder 0.1
  equaliser_mainreservoir 0.06
  equaliservent 0.06
  equaliserventhandleoff 0.1
  equaliserventemergency 0.1
  no3pipevent 1.5
  no3pipe_mainreservoir 0.1
  compressor 5
  trainbrakepipe_reservoir 1
  trainbrakepipevent 0.06
  no3pipe_autobrakecylinder 0.1
  epreservoirpipe_autobrakecylinder 0.1
  mainreservoir_ep 0.1
  vacuumbrakepipe 0.1
  vacuumbrakepipereleasevent 0.1
  vacuumbrakepipevent 0.1
  vacuumbrakereservoir_vacuumbrakepipe 0.1
  vacuumbrakecylinder_vacuumbrakepipe 0.1
  highspeedexhauster_vacuumbrakepipe 0.1
}
volume
{
  scale 1
  trainbrakepipe 0.2
  epreservoirpipe 0.2
  no3pipe 0.2
  no4pipe 0.2
  auxreservoir 0.0384678
  autobrakecylinder 0.00969387
  vacuumbrakepipe 0
  vacuumbrakereservoir 0
  vacuumbrakecylinder 0
  mainreservoir 1.0
  equaliser 0.5
  independantbrakecylinder 0.0103239
}
pressure
{
  scale 1
  compressor 0.00946941
  mainreservoir 0.00946941
  highspeedexhauster 0
  brakepipe 0.00595441
  brakeinitial 0.00560291
  brakefull 0.00398601
  indbrakefull 0.00398601

  trainbrakepipe_start 0.00440781
  epreservoirpipe_start 0
  no3pipe_start 0
  no4pipe_start 0

  auxreservoir_start 0.00504051
  autobrakecylinder_start 0.00489991
  vacuumbrakepipe_start 0
  vacuumbrakereservoir_start 0
  vacuumbrakecylinder_start 0
  mainreservoir_start 0.00876641
  equaliser_start 0.00440781
  independantbrakecylinder_start 0.00489991
}
mass
{
  scale 1
  fuel 6.2156e+006
}
motor
{
  resistance 1.3
  adhesion 2.5
  maxvoltage 600
  maxspeed 21
  brakerratio 55000
  max-accel 1500
  max-decel 5000
  throttle-notches 32
  axle-count 4
  surface-area 150
  moving-friction-coefficient 0.01
  air-drag-coefficient 0.0001
}
throttle-power
{
  0 {
    0 0
  }
  1 {
    0 35
    5 28
    10 18
    12 0
  }
  2 {
    0 85
    5 70
    10 60
    15 30
    30 0
  }
  3 {
    0 140
    5 93
    10 70
    15 62
    30 0
  }
  4 {
    2 187
    5 109
    10 93
    15 87
    30 0
  }
}
```

Continues next page...


```

5 {
    0      281
    5      218
    10     109
    15     87
    30     0
}

6 {
    0      343
    5      265
    10     172
    15     125
    30     0
}

7 {
    0      359
    5      343
    10     187
    15     156
    30     0
}

8 {
    0      436
    3.5    429
    4.25   425
    5      408
    10     234
    15     172
    21     0
}
}
dynamic-brake {
    0 {
        0      0
    }
    1 {
        1.333  0
        2      30
        5      25
        6      15
        7      0
    }
    2 {
        1.333  0
        2      50
        5      35
        7      20
        8      0
    }
    3 {
        1.333  0
        2      60
        5      40
        7      20
        8      0
    }
    4 {
        1.333  0
        3      80
        7      60
        10     20
        12     0
    }
}

5 {
    1.333  0
    5      90
    9      70
    12     25
    15     0
}

6 {
    1.333  0
    5      150
    9      80
    13     70
    17     0
}

7 {
    1.333  0
    5      200
    10     100
    16     60
    19     0
}

8 {
    1.33  0
    5      200
    10     150
    18     50
    21     0
}
}

steam
{
    ; pressure in kPa
    ; flow sizes (nominal figure)
    ; volume in L
    ; mass in kg

    firebox-volume             1000.0
    firebox-to-boiler-heat-flow 0.055
    firebox-efficiency          0.995

    boiler-volume              3000.0
    water-injector-rate        4.0

    westinghouse-volume        100
    main-reservoir-volume      50.0

    cylinder-volume            50.0
    piston-volume-min          1.48
    piston-volume-max          68.7
    piston-area                 0.177
    piston-angular-offsets     0.1
    piston-to-atmosphere-flow  0.0021

    safety-valve-low-pressure  956.0
    safety-valve-low-flow      0.011
    safety-valve-high-pressure 1010.0
    safety-valve-high-flow     0.2

    max-fire-coal-mass         50.0
    max-fire-temperature       1873.0
    shovel-coal-mass           2.0
    burn-rate                   0.0001
    fuel-energy                  100.0

    boiler-to-piston-flow      0.0017
}

```

DIESEL ENGINE FILE BREAKDOWN

kind – asset type

rem – comment used to display engine name

flowsize

rate of flow through pipes, generally leave these settings:

trainbrakepipe 170000
epreservoirpipe 0.1
no3pipe 0.1
no4pipe 0.1
auxreservoirvent 0.1
auxreservoir_no3 0.1
auxreservoir_trainbrakepipe 0.1
autobrakecylindervent 0.1
auxreservoir_autobrakecylinder 0.1
equaliser_mainreservoir 0.06
equaliservent 0.06
equaliserventhandleoff 0.1
equaliserventemergency 0.1
no3pipevent 1.5
no3pipe_mainreservoir 0.1
compressor 10
trainbrakepipe_reservoir 1
trainbrakepipevent 0.06
no3pipe_autobrakecylinder 0.1
epreservoirpipe_autobrakecylinder 0.1
mainreservoir_ep 0.1
vacuumbrakepipe 0.1
vacuumbrakepipereleasevent 0.1
vacuumbrakepipevent 0.1
vacuumbrakereservoir_vacuumbrakepipe 0.1
vacuumbrakecylinder_vacuumbrakepipe 0.1
highspeedexhauster_vacuumbrakepipe 0.1

volume – size of pipes and appliances

scale 1

multiplies volume by given value, generally leave this setting.

trainbrakepipe 0.2
brake pipe volume

epreservoirpipe 0.2
For electro pneumatic braking - not currently in use, generally leave this setting

no3pipe..... 0.2
Independent brake pipe

no4pipe 0.2
Bail pipe - not currently in use, generally leave this setting

Auxreservoir.....0.0384678
Auxiliary reservoir volume.

Autobrakecylinder..... 0.00969387
Brake cylinder volume.

vacuumbrakepipe 0

vacuumbrakereservoir..... 0
vacuumbrakecylinder..... 0
For vacuum braking - not currently in use, generally leave this setting

mainreservoir 0.9
Main reservoir volume.

equaliser 0.5
Equalising reservoir volume

independantbrakecylinder.....0.0103239
Loco brake cylinder volume

pressure

Brake system pressures.

scale 1

Multiplies pressure by given value, generally leave this setting.

Compressor.....0.00946941
(120psi expressed in grams/m³)
Compressor maximum pressure.

mainreservoir.....0.00946941
Main reservoir maximum pressure

highspeedexhauster 0
For vacuum braking - not currently in use, generally leave this setting

brakepipe 0.00736041
(80psi expressed in grams/m³)
Brake pipe pressure when fully charged

brakeinitial.....0.00693861
(72psi expressed in grams/m³)
Brake pipe pressure after initial service reduction (for self lapping brakes)

brakefull 0.0044992
(57psi expressed in grams/m³)
Brake pipe pressure after full service reduction (for self lapping brakes)

indbrakefull 0.005075
Brake cylinder pressure for independant brake service.

trainbrakepipe_start 0.00553261
Brake pipe pressure on loading the game.

epreservoirpipe_start 0
For electro pneumatic braking - not currently in use, generally leave this setting

no3pipe_start 0
no4pipe_start 0
Generally leave these settings.

auxreservoir_start.....0.00553261
 Auxiliary reservoir pressure on loading the game.

autobrakecylinder_start 0.00560291
 Train brake cylinder pressure on loading the game.

vacuumbrakepipe_start 0
vacuumbrakereservoir_start 0
vacuumbrakecylinder_start 0
 For vacuum braking - not currently in use, generally leave this setting

mainreservoir_start 0.00946941
 (100psi expressed in grams/m³)
 Main Reservoir pressure on loading the game.

equaliser_start 0.00553261
 Equalising Reservoir pressure on loading the game.

independantbrakecylinder_start 0.00560291
 Locomotive brake cylinder pressure on loading the game.

mass
scale 1
 Multiplies fuel mass by given value, not currently in use, generally leave this setting.

fuel 6.2156e+006
 Fuel level, not currently in use, generally leave this setting.

motor
resistance.....1.7
 Power figure for DCC, higher resistance value=less power

adhesion 2.5
 Adhesion parameter, higher value=greater adhesion

maxvoltage 600
 Generally leave this setting

maxspeed 40
 Maximum speed for DCC, expressed in metres per second.

Brakeratio..... 55000
 Brake force for pressure reduction

max-accel.....3500
max-decel.....9000
 Parameters for DCC acceleration & deceleration.

axle-count.....4
 Resistance – axle count

surface-area 80
 Resistance – surface area

moving-friction-coefficient.....0.03
 Resistance – moving friction

air-drag-coefficient.....0.00017
 Resistance – air drag

throttle-power

Acceleration variables in cabin mode

1 {		= notch number (1)
0	30	
5	25	= At speed 5, acceleration = 25
10	15	
12	0	
}		

dynamic-brake

Deceleration variables while dynamic braking in cabin mode

1 {		= notch number (1)
1.333	0	
2	30	
5	25	= At speed 5, deceleration = 25
10	15	
12	0	
}		

Equalisation of Pressures There is a point at which no further brake pipe pressure reduction will result in increased braking effort, this is known as full application or equalisation of pressures.

Imagine you made a 26 psi reduction when operating a loco with a 90psi brake pipe. 90psi in the train pipe minus 26psi reduction equals 64 psi in the pipe. Due to the 2.5: 1 ratio of auxiliary reservoir volume to brake cylinder volume, the 26 psi reduction puts 64 psi into the brake cylinder.

As the pressure in the reservoir and the pressure in the cylinder is now equal, no more air will flow into the brake cylinder; and making a further reduction in brake pipe pressure will have no effect on braking.

Equalisation occurs at different pressures, depending on the train pipe feed pressure.

100 psi pipe (e.g. the UK locos - 7 bar) equalisation at 71 psi.

90 psi pipe (e.g. the US locos) equalisation at 64 psi.

72 psi pipe (e.g. French & Queensland locos) equalisation at 49 psi.

The easiest way to set your custom content to the desired brake pipe feed pressure is to copy the entire **pressure** section from the config of a loco that uses the pressure you desire.

Note: Converting PSI to Grams /m cubed...

e.g. 90psi... (90+14.7).0000703

104.7 x .0000703=.00736041

STEAM ENGINE FILE BREAKDOWN

Generally identical to a diesel engine file with the addition of the following:

Units:

Pressure in kPa

Volume in Litres

Mass in kg

Flow sizes (nominal figure)

Temperature degrees Kelvin

steam

firebox-volume..... 1000.0

Physical volume of firebox in Litres.

firebox-to-boiler-heat-flow.....0.055

Rate of heat flow from firebox to boiler and vice-versa.

firebox-efficiency.....0.995

Atmosphere leakage. 1.0 = No leakage.

boiler-volume.....3000.0

Physical volume of boiler in Litres.

water-injector-rate.....4.0

Water injection rate into boiler in Litres/second.

westinghouse-volume.....100

Westinghouse volume in Litres.

piston-volume-min.....1.48

The volume of the space in the cylinder ahead of the piston **at the end** of a full stroke.

piston-volume-max.....68.7

The volume of the space in the cylinder ahead of the piston **at the start** of a full stroke.

piston-area.....0.177

The cross section of one piston in m². It is assumed there is one piston only on each side of the locomotive.

piston-angular-offsets.....0.1

See the additional notes in Comments and New Tags

piston-to-atmosphere-flow.....0.0021

Atmospheric leakage from piston. Nominal hole size.

safety-valve-low-pressure.....956.0

When boiler pressure hits this value in kPa the safety-valve-low-flow release is initiated. (below)

safety-valve-low-flow.....0.011

Lower pressure valve release. Nominal hole size.

safety-valve-high-pressure....1010.0

When boiler pressure hits this value in kPa the safety-valve-high-flow release is initiated. (below)

safety-valve-high-flow.....0.2

Higher pressure valve release. Nominal hole size.

max-fire-coal-mass.....50.0

The maximum mass of coal the firebox can take in kilograms.

max-fire-temperature.....1873.0

Maximum heat obtainable.
(Kelvin scale temperature)

shovel-coal-mass.....2.0

Amount of coal in one shovel load in kilograms.

burn-rate.....0.0001

Coal consumption rate.

fuel-energy.....100.0

Relative energy in kilojoules per kilogram of coal.

boiler-to-piston-flow.....0.0017

Relative energy.

COMMENTS AND NEW TAGS

The above information is specific to the small PB 15 locomotive. A Big Boy locomotive for example, is at the other size extreme for locomotives. The following is an expanded discussion for other size locomotives, of additional tags and options, with some examples.

firebox-volume

westinghouse-volume

main-reservoir-volume

The above tags are currently not implemented.

boiler-volume.....00000.0

maximum-volume.....00000.0

minimum-volume.....00000.0

These above three tags are used together. Because of the way fireboxes are handled in Trainz, basing these figures on the actual litre volume of the boiler results in the water being used too quickly.

All three values for real world boiler size should be multiplied by 10 to give a more realistic consumption rate.

maximum-volume - this volume should be 90% of the boiler-volume, this simulates the steam space left over the top of the water.

minimum-volume - this value should be 90% of the maximum-volume, this represents the working low water level in the boiler.

Alternatively there are three (3) sets of figures that cover all boilers, this is a relatively coarse setting and as such a broad range of vessels can be “blanket” covered:

Small locos with grate areas under 10 Sq metres

<i>boiler-volume</i>	37500.0
<i>minimum-volume</i>	28500.0
<i>maximum-volume</i>	32500.0

Medium locos with a grate area 10 to 25 Sq metres

<i>boiler-volume</i>	47500.0
<i>minimum-volume</i>	37750.0
<i>maximum-volume</i>	40500.0

Large locos with a grate area over 25 Sq metres

<i>boiler-volume</i>	95000.0
<i>minimum-volume</i>	73000.0
<i>maximum-volume</i>	81000.0

initial-boiler-temperature.....380

This line has been added to allow the locomotives to be at an almost ready to go state when the session starts. Then, in conjunction with the *firebox-to-boiler-heat-flow-idle* and *burn-rate-idle* values, it allows a locomotive to stay in this state with the water and fire left for as long as is required, i.e. on stand by. This is nominally 80 - 90% of working pressure.

Full pressure is not required initially, as it only takes a short time to get the fire brightened up, and the boiler to pressure from this point. Most locomotives will go to blowing off pressure with a press of the N key to turn the blower on and a single tap on the space bar for a shovel of coal, remembering that in the real world it is considered very bad practice to have the locomotive blowing the safeties, a waste of resources and energy, and shows a lack of control by the crew.

The following table may be useful as a comparison of initial-boiler-temperature and related pressures. Temperature values (e.g.369) are in degrees Kelvin, see the important Notes to the right of the table:

369	= 869kpa / 140psi	= 460.3K
380	= 1008kpa / 146psi	= 466.5K
381	= 1021kpa / 148psi	= 467.0K
389	= 1122kpa / 163psi	= 471.5K
389.6	= 1129kpa / 165psi	= 471.8K
401	= 1275kpa / 185psi	= 478.2K
404	= 1297kpa / 188psi	= 479.8K
409	= 1374kpa / 199psi	= 482.6K
410	= 1387kpa / 201psi	= 483.2K
411	= 1401kpa / 203psi	= 483.7K
412	= 1410kpa / 204psi	= 484.3K
413	= 1414kpa / 206psi	= 484.8K
424	= 1564kpa / 227psi	= 490.9K
426	= 1589kpa / 230psi	= 492.4K
455	= 1956kpa / 284psi	= 508.2K

piston-angular-offsets

Determines the number of power impulses a locomotive has for each wheel revolution, thus simulating the prototype.

For a 2 cylinder locomotive use:

piston-angular-offsets 0.0174,1.5254,3.0333,4.5413

For a 3 cylinder locomotive use:

piston-angular-offsets 0.0174,1.065,2.107,3.061,4.206,5.253

For a 4 cylinder locomotive use:

piston-angular-offsets 0.0174,0.8028,1.5254,2.3736,3.0333,3.9444,4.5413,5.5152

These are expressed in radians (3.1416 radians = 360 degrees) and the values were determined so as to achieve the smoothest operation possible, they have been deliberately kept away from the sound impulse point. It is recommended not to alter these tested settings.

firebox-to-boiler-heat-flow-idle 0.003

burn-rate-idle 0.003

These two tags in conjunction with the *initial-boiler-temperature* tag are the “brakes” for the boiler when the locomotive is in standby mode, i.e. unattended, and they don’t need any alteration, unless you desire the boiler to run out of water or fire when parked unattended.

See the separate Content Creation Tips and Tricks document for a discussion on how a locomotive using these tags and values may be tuned to give realistic performance.

Notes on the Initial Boiler Temperature Table:

Trainz does not actually use correct Kelvin temperatures for the input to the initial-boiler-temperature tag value, a correction factor is applied to these values in Trainz.

The bolded values in the left column are to be entered in the tag.

The Kelvin values in the right hand column of the table are more representative of the actual Kelvin temperatures associated with the pressures shown.

If you know the initial boiler pressure, for example 1275kpa, enter 401 as the value in the tag.

If you know the initial boiler temperature, for example 467K, enter 381 as the value in the tag.

KIND: BOGEY

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\bogeys

A 'bogey' is a term used for a locomotive or rolling stock wheel mechanism. In some countries these are known as 'Trucks'.

Referenced by the *bogey* tag in a traincar config.txt

Config.txt:

```
kuid <KUID2:###:#####:>
kind bogey
animdist 2.1
mesh-table
{
    default
    {
        mesh Car_bogey.lm
        auto-create 1
    }
    shadow
    {
        mesh Car_bogey_shadow/Car_bogeyshadow.im
    }
}

obsolete-table
{
}

username mybogey
description " "
trainz-build 2.0
category-class AC
category-region-0 AT
category-era-0 1980s

direct-drive 1
```

BREAKDOWN OF CONFIG.TXT

animdist

Leave this tag out if the bogey is not animated.

The distance traveled in meters by the bogey in 1 second (30 frames) of animation. This figure would normally be the circumference of the wheel (multiply the wheel diameter by Pi which is 3.1416). If for instance there are large driving wheels and smaller wheels used on the pilot, of a steam locomotive, you will need to work out the correct value for each bogey using the angular rotation for 30 frames.

For an example calculation see [Page 113](#).

Note: Bogey animations (exported from gmax or 3dsmax) are called "anim.kin".

In previous versions of Trainz, the following lines were used in the config.txt file -

```
anim anim.kin
animation-loop-speed 1.0
```

These are no longer required. Trainz automatically recognises the anim.kin file for the bogey. These lines if used in the config.txt will generate an error log message about incorrectly specified animation being ignored.

Refer to Chapter 4 - Bogeys for modeling guidelines, [Page 112](#).

direct-drive

Steam Locomotive animated driving bogeys only.

Important:

When direct-drive is present, the bogey animation is linked to the steam piston and physics system. If this tag is not included the piston and steam sounds will not work!

Note: The example on the left shows a bogey with LOD mesh reduction. UTC setup (without a mesh-table) will of course still work but we do recommend setting up the config with a mesh-table.

Reversing bogey animation:

*If a front bogey is attached to a train car, and the same bogey is to be used at the rear, but rotated 180 degrees, this is accomplished by entering **bogey-r** in the config.txt file for the train car, for that bogey.*

If the bogey has animation, the animation will have reversed orientation (this will cause bogey animation to play in reverse). The attachment point for the bogey also has to be rotated 180 degrees in 3dmax/gmax to correct the rotation direction.

Refer to [Page 37](#) for description of other config.txt file entries.

KIND: TRAINCAR

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\trains

A 'traincar' is a Locomotive or Rolling stock asset. One of the main features of TRS is that traincars, (namely rolling stock), can now have the ability to transport commodities (products) and to interact with compatible industry assets.

As there are a range of products and product categories, each rolling stock type is set up differently to accommodate the load type both visual representation and through it's config. For this reason we have made available individual downloads for the various rolling stock types on [Page 38](#).

Refer:

KIND: PRODUCT & KIND: PRODUCT-CATEGORY

The following example is a config.txt file from a locomotive. Note the mesh-table and animation setup.

```
kuid <KUID2:####:#####.>
category-class AC
category-region-0 AU
category-era-0 1970s
category-era-1 1980s
category-era-2 1990s
category-era-3 2000s

kuid-table
{
}
obsolete-table
{
}

mesh-table
{
  default
  {
    mesh loco_body/loco_body.lm
    auto-create 1
  }
  shadow
  {
    mesh loco_shadow/loco_shadow.im
  }
  fan1
  {
    mesh loco_body/fan/fan.im
    anim loco_body/fan/fan.kin
    auto-create 1
    att a.fan0
    att-parent default
    animation-loop-speed 1.0
  }
  fan2
  {
    mesh loco_body/fan/fan.im
    anim loco_body/fan/fan.kin
    auto-create 1
    att a.fan2
    att-parent default
    animation-loop-speed 1.0
  }
}
```

```
default-night-forward
{
  mesh loco_body/night/night.im
  auto-create 0
  att a.bog0
  att-parent default
}

bogey <KUID:####:#####>
bogey-1 <KUID:####:#####>
bogey-2-r <KUID:####:#####>
pantograph <KUID:####:#####>
interior <KUID:####:#####>
engine 1
name Electric Loco
mass 37000
company Queensland Rail
origin AU
kind traincar
fonts 2

cabinsway 0

enginespec <KUID:####:#####>
enginesound <KUID:####:#####>
hornsound <KUID:####:#####>

smoke_shade 0.3
smoke_random 2
smoke_slowlife 1
smoke_fastlife 6
smoke_height 7
smoke_fastspeed 4

smoke0
{
  attachment a.exhaust0
  mode speed
  enabled 1
  maxspeedkph 120
  file "lospeed.tfx"
}

smoke1
{
  attachment a.exhaust1
  mode speed
  enabled 1
  maxspeedkph 120
  file "hispeed.tfx"
}

description " "
```


TRAINCAR CONFIG.TXT BREAKDOWN

Some config.txt tags are not explained below. Others are covered in the general config.txt explanation, see [Page 12](#).

default-night-forward

The name for a submesh attached to a locomotive, to show a beam of light for example, in the direction of movement of the locomotive. Trainz recognises the name and turns on the correct mesh depending on the running direction.

Note in the example the use of auto-create 0 to make the mesh invisible when placed. The mesh will be visible when the light switch is activated.

bogey

The bogey KUID number (default for a.bog0 and a.bog1)

bogey-1

The bogey KUID number for a.bog1 (Used only if different to a.bog0)

bogey-r and bogey-1-r

Used instead of 'bogey' and bogey-1. The bogey will have reversed orientation. Note: This will cause bogey animation to play in reverse unless the attachment point for the bogey is also rotated 180 degrees in 3dmax/gmax.

pantograph

The pantograph KUID number inserted at a.pant0, a.pant1, etc. Use this tag only when needed.

interior

Kuid number of the required interior. Inserted at a.cabfront or a.cabback. Use this tag only when needed e.g.locomotives.

engine

States type of traincar.

0 = Rolling stock
1 = Locomotive

mass

Mass in kilograms

company

The Locomotive or car owner

origin

The Country Abbreviation

kind

Traincar

fonts

Indicates how many types of numbering fonts used, e.g.

0 = no fonts used

1 = one font

Digit textures (digit_1.tga to digit_6.tga) replaced automatically with alphanumeric textures (alphanumeric_0 to alphanumeric_9) as numbers are changed via the Surveyor Trains tab - 'Edit Properties' icon (the '?' icon).

2 = two fonts

Digit textures (digit_1a.tga to digit_6a.tga and digit_1b.tga to digit_6b.tga) replaced automatically with alphanumeric textures (alphanumeric_0a to alphanumeric_9a and alphanumeric_0b to alphanumeric_9b) as numbers are changed via the Surveyor Trains tab - 'Edit Properties' icon (the '?' icon).

Enginespec

References the engine KUID number. This specifies the driver physics boundaries for the traincar.

Refer KIND: ENGINE [Page 27](#).

Default location

C:\Program Files\Auran\TRS2004\Engines\

enginesound

References the KUID number for the traincar's sound.

Refer Chapter 3, KIND: ENGINESOUND [Page 39](#).

Default location

C:\Program Files\Auran\TRS2004\World\Custom\enginesound\

hornsound

References the KUID number for the traincar horn sound.

Refer Chapter 3, KIND: HORNSOUND [Page 41](#).

Default location

C:\Program Files\Auran\TRS2004\World\Custom\hornsound\.

cabinsway "strength"

strength is a floating point number. This controls the magnitude of the random roll in internal view. The roll is also affected by the speed of the train. Negative numbers are not used.

0 = no sway

larger numbers (2, 6 etc) = larger sway.

smoke0

Sets boundaries for smoke, steam, vapor and similar effects. Refer: Smoke Effects Chapter 5 on [Page 122](#)

description " "

Description of the model for 'Railyard' information and display on the Download station with the model.

light_color

RGB headlight colour. eg. 255,255,255

This is the color of the train headlight corona, not the lighting itself.

Please download the examples and source files from the links on the next page.

TRAINCAR EXAMPLES

Please download the following rolling stock descriptions and examples for your reference.

COAL HOPPER

A Coal Hopper is a typical example of a rolling stock item that uses an animated load representation. This asset also has animated opening doors and particle effects, each controlled by a script.

Download a zipped pack containing a PDF description, in-game files and source 3dsmax 4 and gmax files of the TRS asset 'Coal hopper'

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Coal_Hopper.zip

TANK CAR

A Tank Car is a typical example of a rolling stock item that can take a liquid load.

Download a zipped pack containing a PDF description, in-game files and source 3dsmax 4 and gmax files of the TRS asset 'GATX Oilco Tank Car'

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Tank_Car.zip

CONTAINER FLATCAR

The Container Flat is a typical example of a rolling stock item that can take a variety of loads by attachment.

Download a zipped pack containing a PDF description, in-game files and source 3dsmax 4 and gmax files of the TRS asset 'Container Flat' Car

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Container_Flat.zip

BOX CAR

The Box Car is a typical example of a rolling stock item that can take the General Goods Product *without* having attachments.

Download a zipped pack containing a PDF description, in-game files and source 3dsmax 4 and gmax files of the TRS asset 'PRR 40ft Box Car'

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Box_Car.zip

PASSENGER STATION AND VEHICLE TUTORIAL

The Passenger Asset Tutorial gives information on passenger stations and vehicles compatible with the new passenger supported features.

Download the Tutorial from:

http://files.auran.com/TRS2004/downloads/contentcreation/SP2-Passenger_Asset_Tutorial.zip

KIND: ENGINESOUND

Default location:

C:\Program Files\Auran\TRS2004\World\custom\enginesound\

These are the locomotive engine sounds, referenced by the *enginesound* tag in a traincar config.txt

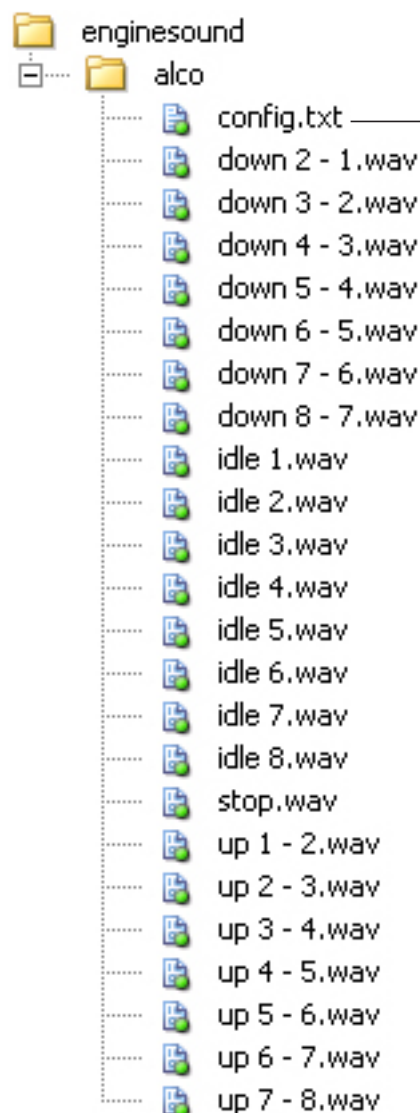
With the inclusion of steam in TRS, we have added additional features for engine sound support.

For diesel and electric loco's the sound is generally as-per the UTC release (described on this page).

Steam sound requirements are described on the following page.

ENGINESOUND - DIESEL AND ELECTRIC

Diesel enginesound Directory Structure



Note:

Diesel enginesound files (.wav) must be located in the same subfolder as the config.txt. You must ensure all custom engine sound files are named the same as those described in the example on the left.

Config.txt (Diesel enginesound)

```
kuid <KUID:###:#####>
kind enginesound
obsolete-table
{
}
category-era-0
category-region-0
category-class
```

Note:

For general information on soundfiles and soundscripts for other uses see [Page 126](#).

ENGINESOUND - STEAM

Custom steam sounds can be created for TRS. The following is a general break down of the required sounds needed and a brief description of how the sounds are used by TRS. These sound files described can be downloaded in zip format from this location: http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Steam_Sound.zip

Steam sounds can be located in the default *enginesound* directory.

These file are the steam engine idling sounds played after the steam engine is stationary for 1, 2 and 3 minutes

- **loco_stationary_fast.wav** (after 1min)
- **loco_stationary_med.wav** (after 2mins)
- **loco_stationary_slow.wav** (after 3mins)

Piston stroke sounds, played every 180 degrees revolution of the piston wheel played in sequence and repeated up to about 40 kph.

- **piston_stroke1.wav**
- **piston_stroke2.wav**
- **piston_stroke3.wav**
- **piston_stroke4.wav**

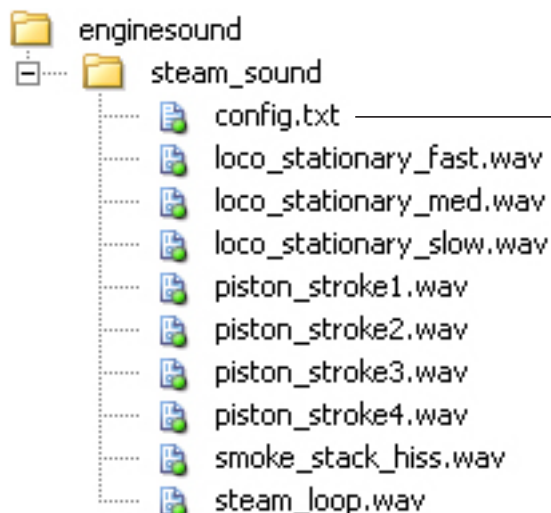
From 40 kph upwards, the following sound loop is cross-faded as the piston sounds die off. The loop is pitched shifted (through code) relative to the locomotive's velocity.

- **steam_loop.wav.**

The general hiss from the smoke stack:

- **smoke_stack_hiss.wav**

Steam enginesound Directory Structure



Config.txt

```
kuid <KUID:###:####>
kind enginesound
obsolete-table
{
}
category-era-0
category-region-0
category-class
```

IMPORTANT Note 1:

You must ensure all custom sound files for steam trains are named the same as those described above.

IMPORTANT Note 2:

The Steam loco driving bogey is connected to the piston and physics system by adding the following tag to the bogey's config.txt: **direct-drive 1**

(See PB_15_bogey2 Config.txt right)

This tag MUST be included for piston and steam sounds to work.

PB_15_bogey2 Config.txt

```
kind bogey
kuid <KUID:44179:50003>
animdist 3.816
category-class AS
category-region-0 AU
category-era-0 1920s
category-era-1 1930s
category-era-2 1940s
category-era-3 1950s
category-era-4 1960s
category-era-5 1970s
category-era-6 1980s
direct-drive 1
```

KIND: HORNSOUND

Default location:

C:\Program Files\Auran\TRS2004\World\custom\hornsound\

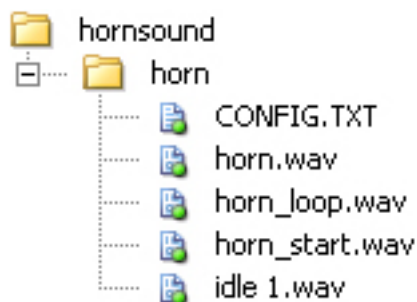
This is the traincar horn sound.

Referenced by the *hornsound* tag in a traincar config.txt

Hornsound Config.txt

```
kind hornsound
kuid <KUID:###:####>
category-era-0
category-region-0
category-class
two-part 1
```

Hornsound Directory Structure



two-part 1

Indicate that the Railyard and Driver hornsounds are different. The Driver hornsound is looping. If this tag is not present, the hornsound defaults to UTC equivalent non-looping format.

Sound files:

- **horn.wav**
'Railyard' hornsound (non-looping)
- **horn_loop.wav**
The looping hornsound used in 'Driver'.
- **horn_start.wav**
The starting sound played before the looping hornsound above.
- **idle 1.wav**
Generally used for the bell sound (bell keystroke = b)

Download DriverCharacter example

An in-game example of this character is available for download through the following link:

<http://www.auran.com/TRS2004/downloads/contentcreation/DriverCharacter.zip>

We suggest you re-skin the mesh files provided in this zip in order to keep consistency.

KIND: DRIVERCHARACTER

Default location:

C:\Program Files\Auran\TRS2004\World\custom\TEXTURES\

This is the TRS locomotive driver character.

DriverCharacter Config.txt

dave64.tga

```
kind DriverCharacter
face-texture dave64.texture
KUID <KUID:16:10129>
mesh <KUID:-3:10130>
```



DriverCharacter Directory Structure



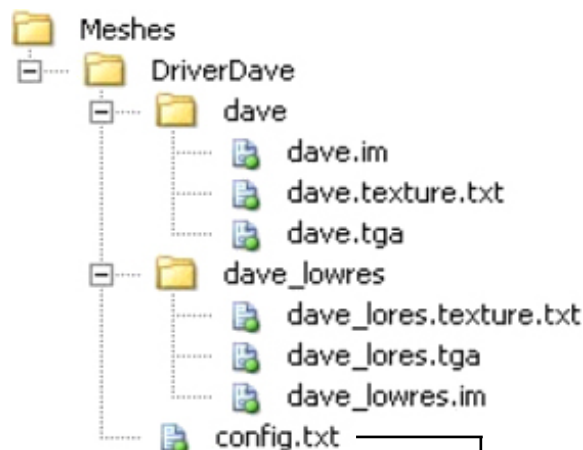
face-texture

This is the driver icon used in TRS.

mesh

This refers to the kuid of the mesh inserted in the locomotive mesh at a.driver0, (when in the Driver Module).

DriverCharacter Mesh Directory Structure



DriverCharacter Mesh Config.txt

```
kind mesh
KUID <KUID:-3:10130>
mesh-table
{
    standing
    {
        mesh dave/dave.im
    }
    sitting
    {
        mesh dave_lowres/dave_lowres.im
    }
}
```

KIND: INTERIOR

Default location:

C:\Program Files\Auran\TRS2004\World\custom\interiors\

This is the traincar interior.

Referenced by the *interior* tag in a traincar config.txt

In TRS, an interior config.txt file now has the ability to be setup using a mesh-table. This gives greater control over animations and gives the ground work for script implementation.

In the following example, the generic UP DD40 interior has scripted, animated wipers and a fan, both controlled by a switches. Also, when these switches are in the 'on' position, a mesh is rendered to represent the switch light coming on.

Actually, the DD40 interior is a great example of what can be done using standard levers alone. The sliding windows, the retractable sun visors and the swivel chair are all 'levers'. Sure they contain no real function... It adds a bit of fun if anything! ☺

Download DD40 Interior source and in-games files here:

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_dd40_interior.zip

Remember: Interiors created using a mesh-table cannot be used in pre-TRS versions of Trainz.

DD40 Interior Config.txt

```
kind interior
```

```
script "DD40Cabin"  
class "DD40Cabin"
```

```
kuid <KUID:-3:10085>
```

```
cameralist
```

```
{  
  camera0 -0.564, 0.984, 0.987, 13.528, -0.137  
  camera1 0.707, -0.333, 1.023, -6.189, -0.049  
  camera2 1.317, 0.949, 1.023, -13.501, -0.122  
  camera3 -1.058, 0.522, 0.887, 0.1, -0.089  
  camera4 -1.382, 0.78, 0.987, 6.592, -0.194  
  camera5 -1.361, 0.923, 1.137, 6.902, -0.503  
  camera6 -0.618, 0.715, 1.023, -0.983, -0.234  
}
```

```
cameradefault 3
```

```
mesh-table
```

```
{  
  default  
  {  
    mesh gen_dd40_int.im  
    auto-create 1  
  }  
}
```

```
Fan_Switch
```

```
{  
  kind lever  
  auto-create 1  
  mesh switch_red.im  
  att a.switch0
```

DD40 Cab interior



```
limits 0, 1  
angles 0, -2  
notches 0, 1  
notchheight 0, 0  
mousespeed -1  
radius 0.05  
att-parent default  
}  
switchlight0  
{  
  kind light  
  mesh switchlight.pm  
  att a.switch0  
  auto-create 0  
  att-parent default  
}  
Wiper_Switch  
{  
  kind lever  
  auto-create 1  
  mesh switch_red.im  
  att a.switch1  
  limits 0, 1  
  angles 0, -2  
  notches 0, 1  
  notchheight 0, 0  
  mousespeed -1  
  radius 0.05  
  att-parent default  
}  
switchlight1  
{  
  kind light  
  mesh switchlight.pm  
  att a.switch1  
  auto-create 0  
  att-parent default  
}  
Switch_3  
{  
  kind lever  
  auto-create 1  
  mesh switch.pm  
  att a.switch2  
  limits 0, 1  
  angles 0, -2  
  notches 0, 1  
  notchheight 0, 0
```

Continues next page...


```

    mousespeed -1
    radius 0.05
    att-parent default
}
switchlight2
{
    kind light
    mesh switchlight.pm
    att a.switch2
    auto-create 0
    att-parent default
}
Switch_4
{
    kind lever
    auto-create 1
    mesh switch.pm
    att a.switch3
    limits 0, 1
    angles 0, -2
    notches 0, 1
    notchheight 0, 0
    mousespeed -1
    radius 0.05
    att-parent default
}
switchlight3
{
    kind light
    mesh switchlight.pm
    att a.switch3
    auto-create 0
    att-parent default
}
Switch_5
{
    kind lever
    auto-create 1
    mesh switch.pm
    att a.switch4
    limits 0, 1
    angles 0, -2
    notches 0, 1
    notchheight 0, 0
    mousespeed -1
    radius 0.05
    att-parent default
}
switchlight4
{
    kind light
    mesh switchlight.pm
    att a.switch4
    auto-create 0
    att-parent default
}
Switch_6
{
    kind lever
    auto-create 1
    mesh switch.pm
    att a.switch5
    limits 0, 1
    angles 0, -2
    notches 0, 1
    notchheight 0, 0
    mousespeed -1
    radius 0.05
    att-parent default
}
}
switchlight5
{
    kind light
    mesh switchlight.pm
    att a.switch5
    auto-create 0
    att-parent default
}
Switch_7
{
    kind lever
    auto-create 1
    mesh switch.pm
    att a.switch6
    limits 0, 1
    angles 0, -2
    notches 0, 1
    notchheight 0, 0
    mousespeed -1
    radius 0.05
    att-parent default
}
switchlight6
{
    kind light
    mesh switchlight.pm
    att a.switch6
    auto-create 0
    att-parent default
}
Switch_8
{
    kind lever
    auto-create 1
    mesh switch.pm
    att a.switch7
    limits 0, 1
    angles 0, -2
    notches 0, 1
    notchheight 0, 0
    mousespeed -1
    radius 0.05
    att-parent default
}
switchlight7
{
    kind light
    mesh switchlight.pm
    att a.switch7
    auto-create 0
    att-parent default
}
reverser_lever
{
    kind lever
    auto-create 1
    mesh reverser_lever.pm
    att a.reverser_lever
    limits 0, 2
    angles 0.55, -0.55
    notches 0, 0.5, 1
    notchheight 1, 1, 1
    att-parent default
}
independantbrake_lever
{
    kind lever

```

Continues next page...


```

    auto-create 1
    mesh ind_brake_lever.pm
    att a.ind_brake_lever
    limits 0, 32
    angles 0, -2.1
    notches 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
0.7, 0.8, 0.9, 1
    notchheight 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1
    radius 0.15
    att-parent default
}
trainbrake_lever
{
    kind lever
    auto-create 1
    mesh train_brake_lever.pm
    att a.train_brake_lever
    limits 0, 4
    angles 0, -2.4
    notches 0, 0.25, 0.27, 0.29, 0.31, 0.33,
0.35, 0.37, 0.39, 0.41, 0.43, 0.45, 0.47, 0.49,
0.5, 0.75, 1
    notchheight 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 1, 1, 1
    radius 0.15
    att-parent default
}
throttle_lever
{
    kind lever
    auto-create 1
    mesh throttle_lever.pm
    att a.throttle_lever
    limits 0, 8
    angles 1.2, 0
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 1, 2, 2, 2, 2, 2, 2, 2, 1
    radius 0.35
    att-parent default
}
dynamicbrake_lever
{
    kind lever
    auto-create 1
    mesh dynamic_lever.pm
    att a.dynamic_brake
    limits 0, 2
    angles 0, -1.2
    notches 0, 0.5, 1
    notchheight 1, 1, 1
    radius 0.35
    att-parent default
}
light_switch
{
    kind lever
    auto-create 1
    att a.light_switch
    mesh dial.pm
    limits 0, 1
    angles 0, 4.5
    notches 0, 0.5, 1
    notchheight 0, 0, 0
    att-parent default
}
ampmeter_needle
{
    kind needle
    auto-create 1

```

```

    mesh ampmeter_needle.pm
    att a.ammeter
    limits 0, 1500
    angles 0, 1.9
    att-parent default
}
bploco_equaliser
{
    kind needle
    auto-create 1
    mesh whitepress_needle.pm
    att a.equaliser_pressure
    limits 0, 1000
    att-parent default
}
bplocomain_needle
{
    kind needle
    auto-create 1
    mesh redpress_needle.pm
    att a.main_res_pressure
    limits 0, 1000
    att-parent default
}
bptrainbrakecylinder_needle
{
    kind needle
    auto-create 1
    mesh redpress_needle.pm
    att a.brake_cyl_pressure
    limits 0, 1000
    att-parent default
}
bptrainbrakepipe_needle
{
    kind needle
    auto-create 1
    mesh whitepress_needle.pm
    att a.brake_pipe_pressure
    limits 0, 1000
    att-parent default
}
speedo_needle
{
    kind needle
    auto-create 1
    mesh whitepress_needle.pm
    att a.speedo
    limits 0, 48
    att-parent default
}
horn
{
    kind lever
    auto-create 1
    mesh horn.pm
    att a.horn
    limits 0, 1
    angles 0, -0.6
    notches 0, 1
    notchheight 3, 3
    radius 0.16
    mousespeed -2
    att-parent default
}
wheelslip_light
{
    kind light
    auto-create 0
    mesh wheelslip.pm

```

Continues next page...

```

}
swivel_chair
{
    kind lever
    auto-create 1
    mesh chair.im
    att a.chair1
    limits 0, 8
    angles 6.8, -6.8
    radius 0.5
    mousespeed 0.2
    att-parent default
}
dial0
{
    kind lever
    auto-create 1
    att a.dial0
    mesh dial.pm
    limits 0, 1
    angles 0, 3.8
    notches 0, 1
    notchheight 0, 0
    att-parent default
}
dial1
{
    kind lever
    auto-create 1
    att a.dial1
    mesh dial.pm
    limits 0, 1
    angles 0, 3.8
    notches 0, 1
    notchheight 0, 0
    att-parent default
}
fan
{
    mesh fan.im
    anim fan.kin
    auto-create 1
    att a.fan
    att-parent default
}
visor0
{
    mesh visor.im
    auto-create 1
    att a.visor0
    att-parent default
    kind lever
    mousespeed -1
    limits 0, 1
    angles 0, 1.7
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
}
visor1
{
    mesh visor.im
    auto-create 1
    att a.visor1
    att-parent default
    kind lever
    mousespeed -1
    limits 0, 1
    angles 0, 1.7
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
    kind lever
}
sl_wind_R_Fr
{
    mesh sliding_wind_R_Fr.im
    auto-create 1
    att a.sliding_wind_R_Fr
    att-parent default
    limits 0, 1.0
    angles 0, 0.011
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
    kind lever
}
sl_wind_R_Bk
{
    mesh sliding_wind_R_Bk.im
    auto-create 1
    att a.sliding_wind_R_Bk
    att-parent default
    limits 0, 1.0
    angles 0, -0.011
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
    mousespeed -1
    kind lever
}
sl_wind_L_Fr
{
    mesh sliding_wind_L_Fr.im
    auto-create 1
    att a.sliding_wind_L_Fr
    att-parent default
    limits 0, 1.0
    angles 0, -0.011
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
    kind lever
}
sl_wind_L_Bk.im
{
    mesh sliding_wind_L_Bk.im
    auto-create 1
    att a.sliding_wind_L_Bk
    att-parent default
    limits 0, 1.0
    angles 0, 0.011
    notches 0, 0.125, 0.25, 0.375, 0.5, 0.625,
0.75, 0.875, 1
    notchheight 0, 0, 0, 0, 0, 0, 0, 0, 0
    mousespeed -1
    kind lever
}
wipers
{
    mesh wipers.im
    anim wipers.kin
    auto-create 1
    att a.wipers
    att-parent default
}
}

```

End of DD40 interior config.txt

DD40Cabin.gs

This is the DD40 interior script file.

This sets up the fan and wiper animations to be switch controlled and controls the visibility of switchlights.

```
include "defaultlocomotivecabin.gs"

class DD40CabinData isclass CabinData
{
    public bool animatingFan;
    public bool animatingWiper;
    public bool switchOn3;
    public bool switchOn4;
    public bool switchOn5;
    public bool switchOn6;
    public bool switchOn7;
    public bool switchOn8;
};

class DD40Cabin isclass
DefaultLocomotiveCabin
{
    // Switches
    CabinControl cabin_fan_switch;
    CabinControl window_wipers;
    CabinControl switch3;
    CabinControl switch4;
    CabinControl switch5;
    CabinControl switch6;
    CabinControl switch7;
    CabinControl switch8;

    // Lights
    CabinControl cabin_fan_light;
    CabinControl window_wipers_light;
    CabinControl light3;
    CabinControl light4;
    CabinControl light5;
    CabinControl light6;
    CabinControl light7;
    CabinControl light8;

    thread void SlowFanDown(void);
    thread void SpeedFanUp(void);

    void UpdateFan(void);
    void UpdateWipers(void);
    thread void RunAnimation(void);

    float fanSpeed;
    bool isFanSpeedingUp;
    bool isFanSlowingDown;

    //! Attach this cabin to a game object
    (i.e. a locomotive).
    //
    // Param: obj Game object to attach this
    cabin to (usually a Locomotive).
    //
    void Attach(GameObject obj)
    {
        inherited(obj);

        // get cabin data
        CabinData cd = loco.GetCabinData();
        if(cd)
        {
            // reset the controls from saved values
            DD40CabinData ddc =
```

```
cast<DD40CabinData>cd;
            float value = 0.0;

            // ANIMATING FAN
            if (ddcd.animatingFan)
            {
                fanSpeed = 1.0;
                SetMeshAnimationSpeed("fan", 1.0);
                StartMeshAnimationLoop("fan");
                value = 1.0;
            }
            else
            value = 0.0;

            cabin_fan_switch.SetValue(value);
            cabin_fan_light.SetValue(value);

            // ANIMATING WIPER
            if (ddcd.animatingWiper)
            {
                value = 1.0;
                SetMeshAnimationSpeed("wipers", 1.0);
                StartMeshAnimationLoop("wipers");
            }
            else
            value = 0.0;

            window_wipers.SetValue(value);
            window_wipers_light.SetValue(value);

            // SWITCH 3
            if (ddcd.switchOn3)
            value = 1.0;
            else
            value = 0.0;
            switch3.SetValue(value);
            light3.SetValue(value);

            // SWITCH 4
            if (ddcd.switchOn4)
            value = 1.0;
            else
            value = 0.0;
            switch4.SetValue(value);
            light4.SetValue(value);

            // SWITCH 5
            if (ddcd.switchOn5)
            value = 1.0;
            else
            value = 0.0;
            switch5.SetValue(value);
            light5.SetValue(value);

            // SWITCH 6
            if (ddcd.switchOn6)
            value = 1.0;
            else
            value = 0.0;
            switch6.SetValue(value);
            light6.SetValue(value);

            // SWITCH 7
            if (ddcd.switchOn7)
            value = 1.0;
            else
            value = 0.0;
            switch7.SetValue(value);
            light7.SetValue(value);
```

Continues next page...

```

// SWITCH 8
if (ddcd.switchOn8)
value = 1.0;
else
value = 0.0;
switch8.SetValue(value);
light8.SetValue(value);
}
else
{
DD40CabinData ddd = new
DD40CabinData();
loco.SetCabinData(ddd);
}
}

void UserPressKey(string s)
{
if(s == "cabin-fans")
{
DD40CabinData cd = cast<DD40CabinData>1
oco.GetCabinData();
cd.animatingFan = !cd.animatingFan;

float value;
if (cd.animatingFan)
value = 1.0;
else
value = 0.0;

cabin_fan_switch.SetValue(value);
cabin_fan_light.SetValue(value);

UpdateFan();
}
if(s == "wipers")
{
DD40CabinData cd = cast<DD40CabinData>1
oco.GetCabinData();
cd.animatingWiper = !cd.animatingWiper;

float value;
if (cd.animatingWiper)
value = 1.0;
else
value = 0.0;

window_wipers.SetValue(value);
window_wipers_light.SetValue(value);

UpdateWipers();
}
}

public void Init(void)
{
inherited();

cabin_fan_switch = GetNamedControl("fan_
switch");
window_wipers = GetNamedControl("wiper_
switch");
switch3 = GetNamedControl("switch_3");
switch4 = GetNamedControl("switch_4");
switch5 = GetNamedControl("switch_5");
switch6 = GetNamedControl("switch_6");
switch7 = GetNamedControl("switch_7");
switch8 = GetNamedControl("switch_8");

```

```

cabin_fan_light = GetNamedControl("switch
light0");
window_wipers_light = GetNamedControl("sw
itchlight1");
light3 = GetNamedControl("switchlight2");
light4 = GetNamedControl("switchlight3");
light5 = GetNamedControl("switchlight4");
light6 = GetNamedControl("switchlight5");
light7 = GetNamedControl("switchlight6");
light8 = GetNamedControl("switchlight7");

RunAnimation();

}

void UserSetControl(CabinControl p_control,
float p_value)
{
DD40CabinData cd = cast<DD40CabinData>loc
o.GetCabinData();

if (p_control == cabin_fan_switch)
{
bool wantFanAnimation = p_value > 0.5;
if (wantFanAnimation !=
cd.animatingFan)
{
cd.animatingFan = wantFanAnimation;
UpdateFan();
}

if (wantFanAnimation)
cabin_fan_light.SetValue(1.0);
else
cabin_fan_light.SetValue(0.0);
}

else if (p_control == window_wipers)
{
bool wantWiperAnimation = p_value >
0.5;
if (wantWiperAnimation !=
cd.animatingWiper)
{
cd.animatingWiper =
wantWiperAnimation;
UpdateWipers();
}

if (wantWiperAnimation)
window_wipers_light.SetValue(1.0);
else
window_wipers_light.SetValue(0.0);
}

else if (p_control == switch3)
{
bool isOn = p_value > 0.5;
float value = 0.0;
if (isOn)
value = 1.0;

light3.SetValue(value);
cd.switchOn3 = value;

```

Continues next page...

```

    }
    else if (p_control == switch4)
    {
        bool isOn = p_value > 0.5;
        float value = 0.0;
        if (isOn)
            value = 1.0;

        light4.SetValue(value);
        cd.switchOn4 = value;
    }

    else if (p_control == switch5)
    {
        bool isOn = p_value > 0.5;
        float value = 0.0;
        if (isOn)
            value = 1.0;

        light5.SetValue(value);
        cd.switchOn5 = value;
    }

    else if (p_control == switch6)
    {
        bool isOn = p_value > 0.5;
        float value = 0.0;
        if (isOn)
            value = 1.0;

        light6.SetValue(value);
        cd.switchOn6 = value;
    }

    else if (p_control == switch7)
    {
        bool isOn = p_value > 0.5;
        float value = 0.0;
        if (isOn)
            value = 1.0;

        light7.SetValue(value);
        cd.switchOn7 = value;
    }

    else if (p_control == switch8)
    {
        bool isOn = p_value > 0.5;
        float value = 0.0;
        if (isOn)
            value = 1.0;

        light8.SetValue(value);
        cd.switchOn8 = value;
    }

    else
        inherited(p_control, p_value);
}

thread void SlowFanDown(void)
{
    DD40CabinData cd = cast<DD40CabinData>loc
o.GetCabinData();

    if (isFanSlowingDown)
        return;

    isFanSlowingDown = true;
    isFanSpeedingUp = false;

    // Slow it down...
    while (fanSpeed > 0.1)
    {
        fanSpeed = fanSpeed - 0.1;
        SetMeshAnimationSpeed("fan", fanSpeed);

        Sleep(0.5);
        if (!isFanSlowingDown)
            return;
    }

    fanSpeed = 0.0;
    StopMeshAnimation(vfan");
}

thread void SpeedFanUp(void)
{
    DD40CabinData cd = cast<DD40CabinData>loc
o.GetCabinData();

    if (isFanSpeedingUp)
        return;

    isFanSpeedingUp = true;
    isFanSlowingDown = false;

    // Speed it up...
    while (fanSpeed < 1.0)
    {
        fanSpeed = fanSpeed + 0.1;
        SetMeshAnimationSpeed("fan", fanSpeed);
        StartMeshAnimationLoop("fan");

        Sleep(0.5);
        if (!isFanSpeedingUp)
            return;
    }

    fanSpeed = 1.0;
    SetMeshAnimationSpeed("fan", fanSpeed);
    //isFanSpeedingUp = false;
}

void UpdateFan(void)
{
    DD40CabinData cd = cast<DD40CabinData>loc
o.GetCabinData();

    if (cd.animatingFan)
        SpeedFanUp();
    else
        SlowFanDown();
}

void UpdateWipers(void)
{
    DD40CabinData cd = cast<DD40CabinData>loc
o.GetCabinData();

    // Don't need to worry about else, as it
    // will be handled when the loop is done.
    if (cd.animatingWiper)
    {
        SetMeshAnimationSpeed("wipers", 1.0);
        StartMeshAnimationLoop("wipers");
    }
}

```

Continues next page...

```

}

thread void RunAnimation(void)
{
    DD40CabinData cd = cast<DD40CabinData>loco.GetCabinData();

    wait()
    {
        on "Animation-Event", "wiperstop":
        if (!cd.animatingWiper)
        StopMeshAnimation("wipers");
        continue;
    }
}
};

```

INTERIOR CONFIG.TXT BREAKDOWN

script

class

This refers to the name of the script file and the class of asset it is (the class must match that stated within the script file).

cameralist

Multiple in-cab camera positions relative to a.cabfront.

0,0,0,0,0 =left/right, front/back, up/down, yaw, pitch

*To determine these variables add **-freeintcam** to the trainzoptions.txt. Pan around the interior using arrow keys and mouse. Co-ordinates are displayed at bottom-left of screen. Make sure you include the negative sign for coordinates where appropriate.*

cameradefault

The in-cab camera view Trainz defaults to when entering the cab.

INTERIOR ATTACHMENT TYPES

pantograph_lever

Pantograph lever/switch. For raising and lowering pantographs on electric locos.

horn

Locomotive's horn.

independantbrake_lever

Independent (Loco) brake lever

reverser_lever

Reverser lever (Forward/Neutral/Reverse)

throttle_lever

Throttle / power handle

trainbrake_lever

Train brake lever - self lapping

trainbrakelap_lever

Train brake lever with lap position.

dynamicbrake_lever

For selecting dynamic brake

bplocomain_needle

Main reservoir pressure needle

bploco_equalizer

Equalising reservoir pressure needle

bptrainbrakepipe_needle

Brake pipe pressure needle

bptrainbrakecylinder_needle

Brake cylinder pressure needle

speedo_needle

Speedometer needle

ampmeter_needle

Power meter needle

flow_needle

Flow gauge needle

windows

Textured mesh with low opacity (semi-transparent) to give impression of reflection. This mesh has the same 3D origin point as the main .pm model, therefore does not require an attachment point

wheelslip_light

A warning light mesh that is only visible when the locomotive loses traction. This mesh has the same 3D origin point as the main .pm model, therefore does not require an attachment point

switch0, switch1 etc

Switches

light_switch

Headlight switch

INTERIOR ATTACHMENT VARIABLES

Kinds: **lever**

Levers, switches, dials etc

animated-lever

Animated Levers etc e.g. in steam cabs

collision-proxy

Mouse collisions for animated levers

needle

Gauge needles (i.e. Speedo, brake pres.)

pullrope

Pull rope horn as in the F7

light

Wheelslip light

mesh

Mesh file to be inserted.

att

Attachment point where mesh is inserted. If no attachment point is specified the mesh will be inserted at a.cabfront (the same insertion point as main mesh)

limits

Mathematical boundaries Trainz uses to determine the objects function. These values vary as different objects use different mathematical units. Generally use the default values used in the config files provided.

angles

Rotational boundaries in radians relative to its attachment point. Refer to the radian/degree circle diagram below to help you out.

notches

The position of notches within the angle boundaries. These are represented as decimal points between and including 0 and 1.

notchheight

The size of the notches specified.

radius

The notch position relative to the attachment point.

mousespeed

This controls the use of the mouse on screen. Use this to control the mouse speed and push/pull direction for levers and dials.

- `mousespeed -1` Inverts mouse direction.
- `mousespeed 2` Doubles mouse speed in default direction.
- `mousespeed -0.5` Inverts mouse direction and halves the speed.

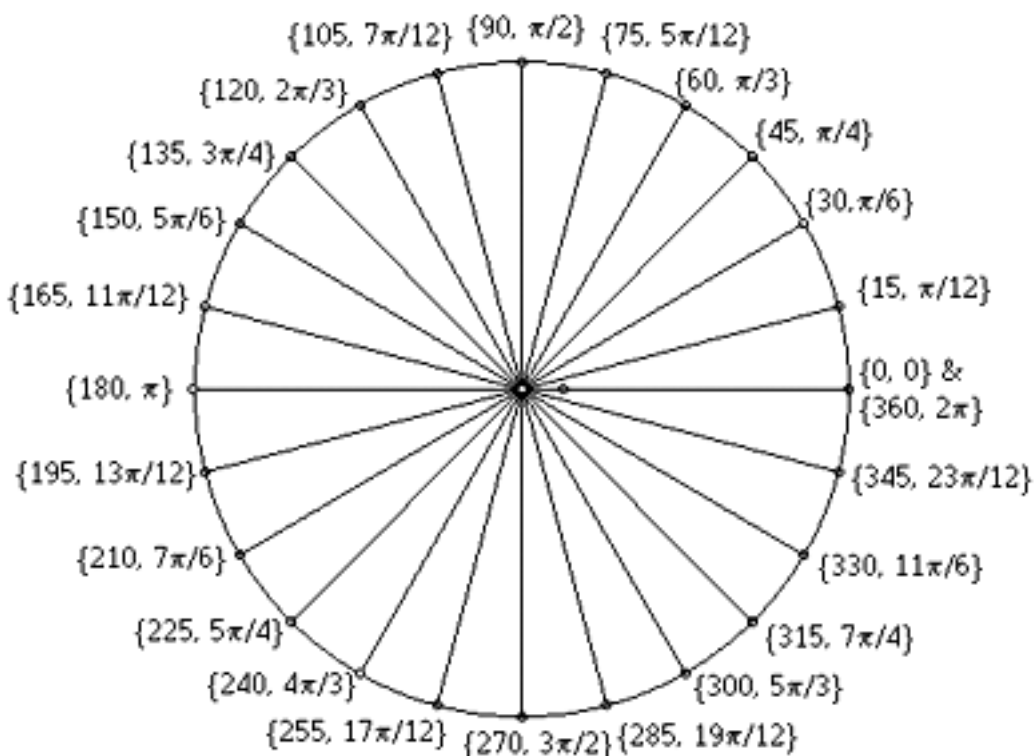
test-collisions 0

Mouse cannot be used for this mesh. Collision mesh used instead. i.e. animated-levers

opacity

Usually used for the window mesh to give transparency (and the impression of reflection).

RADIAN / DEGREE CIRCLE (for angle reference)



STEAM CAB INTERIORS

Overview

TRS steam cab interiors have been set-up in generally the same way as diesel and electric cabs with a few additional steam specific features.

Many of the levers and fireplates have several moving objects and required mouse controlled animations. This differed from the usual lever types with only one object, set to rotate around an attachment point.

Not only did the levers need reviewing, but the cab firebox itself had to produce fire and glow variations and the coal shoveller needed to be controlled and linked to the coal requirements also.

Download PB15 Interior source and in-games files here:

http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_PB15_interior.zip

See *PB15 interior Config.txt* on the following page.

```
fan
{
    mesh fan/fan.im
    anim fan/fan.kin
    auto-create 1
    att a.fan
    att-parent default
}
```

PB15 Steam Interior Config.txt

```
kuid <KUID:-3:10191>
kind interior

script "pb15cabin"
class "PB15Cabin"

camera -0.769, 0.566, 0.617
cameralist
{
    camera0 0.817, 0.026, 0.654, 0.026, -0.146
    camera1 -0.808, 0.474, 0.694, 0.214, -0.773
    camera2 -1.429, 0.461, 0.617, 0.162, -0.075
    camera3 -0.769, 0.566, 0.617, 0.157, -0.125
    camera4 0.703, 0.831, 0.694, -0.715, -0.561
    camera5 1.344, 0.305, 0.617, -0.015, -0.049
}
cameradefault 3

obsolete-table
{
    0 <KUID:44179:55003>
}

soundscript
{
    coal_on
    {
        trigger coal_on
        attachment a.coalman
        ambient 1
        nostartdelay 1
        repeat-delay 0,0.001
        distance 5,200
        sound
        {
            sound/coal_shovel2.wav
        }
    }
    shovel_againstmetal
    {
        trigger shovel_againstmetal
        attachment a.coalman
        ambient 1
        nostartdelay 1
        repeat-delay 0,0.001
        distance 5,200
        sound
        {
            sound/shovel_hit.wav
        }
    }
    step_metal
    {
        trigger step_metal
        attachment a.coalman
        ambient 1
        nostartdelay 1
        repeat-delay 0,0.001
        distance 5,200
        sound
        {
            sound/metal_footstep_left.wav
        }
    }
    coal_off
    {
        trigger coal_off
        attachment a.coalman
        ambient 1
        nostartdelay 1
        repeat-delay 0,0.001
        distance 5,200
        sound
        {
            sound/coal_into_firebox1.wav
        }
    }
}

mesh-table
{
    default
    {
        mesh PB_interior_main.im
        auto-create 1
    }

    trainbrake_lever
    {
        mesh brake_lever/brake_lever.im
        auto-create 1
        att a.brake
        limits 0, 4
        angles -0.75, 0.35
        notches 0, 0.5, 1.0
        notchheight 1, 1, 1
        radius 0.16
        att-parent default
        kind lever
    }

    left_window
    {
        mesh window_sides.im
        auto-create 1
        limits 0, 1.0
        angles 0, -1
        notches 0, 1.0
        notchheight 1, 1
        att a.window_l
        att-parent default
        kind lever
    }

    right_window
    {
        mesh window_sides.im
        auto-create 1
        att a.window_r
        att-parent default
        limits 0, 1.0
        angles 0, -1
        notches 0, 1.0
        notchheight 1, 1
        kind lever
    }

    left_sliding_window
    {
        mesh window_sliding.im
        auto-create 1
        att a.windowsliding_l
        att-parent default
        limits 0, 1.0
        angles 0, -0.009
        notches 0, 1.0
        notchheight 1, 1
        kind lever
    }
}
```

Continues next page...

```

}

right_sliding_window
{
    mesh window_sliding.im
    auto-create 1
    att a.windowsliding_r
    att-parent default
    limits 0, 1
    angles 0, -0.009
    notches 0, 1.0
    notchheight 1,1
    kind lever
}

cylinder_drain
{
    limits 0,4
    angles 0,-0.3
    mesh cylinder_clean/cylinder_clean.im
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    att a.cylinderclean
    att-parent default
    kind lever
}

regulator
{
    mesh regulator/regulator.im
    anim regulator/regulator.kin
    auto-create 1
    att a.regulator
    limits 0, 1.0
    x-notches 0, 0.25, 0.5, 0.75, 1.0
    x-notchheight 1, 1, 1, 1, 1
    att-parent default
    kind animated-lever
    test-collisions 0
    mousespeed -1.0
}

regulator-collision-box
{
    mesh regulator/selection_box/selection_
box.im
    att-parent regulator
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent regulator
}

seat0
{
    mesh seat/seat.im
    anim seat/seat.kin
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    att a.seat0
    att-parent default
    kind animated-lever
    test-collisions 0
}

seat0-collision-box
{
    mesh seat/selection_box/selection_box.im
    att-parent seat0
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent seat0
}

seat1
{
    mesh seat/seat.im
    anim seat/seat.kin
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    att a.seat1
    att-parent default
    kind animated-lever
    test-collisions 0
}

seat1-collision-box
{
    mesh seat/selection_box/selection_box.im
    att-parent seat1
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent seat1
}

water_injector_0
{
    mesh injector/injector.im
    anim injector/injector.kin
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    att a.injector0
    att-parent default
    kind animated-lever
    test-collisions 0
    mousespeed -1.0
}

water_injector_0-collision-box
{
    mesh injector/selection_box/selection_
box.im
    att-parent water_injector_0
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent water_injector_0
}

water_injector_1
{
    mesh injector/injector.im
    anim injector/injector.kin
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0

```

Continues next page...

```

    notchheight 1, 1
    att a.injector1
    att-parent default
    kind animated-lever
    test-collisions 0
    mousespeed -1.0
}

water_injector_1-collision-box
{
    mesh injector/selection_box/selection_
box.im
    att-parent water_injector_1
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent water_injector_1
}

fire_plates
{
    mesh fireplates/fireplates.im
    anim fireplates/fireplates.kin
    auto-create 1
    kind animated-lever
    test-collisions 0
    notches 0, 1.0
    notchheight 1,1
    limits 0, 1.0
}

fire_plates-collision-box
{
    mesh fireplates/selection_box/selection_
box.im
    att-parent fire_plates
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent fire_plates
}

water_valve0
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve0
    att-parent default
    kind lever
}

water_valve1
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve1
    att-parent default
    kind lever
}

water_valve2
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve2
    att-parent default
    kind lever
}

water_valve3
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve3
    att-parent default
    kind lever
}

water_valve4
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve4
    att-parent default
    kind lever
}

water_valve5
{
    mesh water_valve/water_valve.im
    auto-create 1
    att a.water_valve5
    att-parent default
    kind lever
}

blowdown
{
    kind lever
    mesh blowdown/blowdown.im
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    angles 0, 0.01
    att a.blowdown
    att-parent default
    mousespeed -1.0
}

sanding_lever
{
    kind lever
    mesh sanding_lever/sanding_lever.im
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    angles 0, 0.2
    att a.sanding_lever
    att-parent default
    mousespeed -1.0
}

whistle_lever
{
    kind lever
    mesh whistle_lever/whistle_lever.im
    auto-create 1
    limits 0, 1.0
    notches 0, 1.0
    notchheight 1, 1
    angles 0, 0.3
    att a.whistle
    att-parent default
}

bplocomain_needle
{

```

Continues next page...

```

    kind needle
    mesh main_res_needle/main_res_needle.im
    auto-create 1
    att a.reservoir_press_red
    att-parent default
    limits 0, 3600
}

bptrainbrakepipe_needle
{
    kind needle
    mesh trainline_needle_black/trainline_
needle_black.im
    auto-create 1
    att a.brake_press_black
    att-parent default
    limits 0, 3600
}

boiler_needle
{
    kind needle
    mesh boiler_needle/boiler_needle.im
    att a.boiler_pressure
    limits 0, 1570
    auto-create 1
}

reverser
{
    mesh reverser/reverser.pm
    anim reverser/reverser.kin
    auto-create 1
    kind animated-lever
    test-collisions 0
    limits -1.0, 1.0
}

reverser-collision-box
{
    mesh reverser/selection_box/selection_
box.im
    att-parent reverser
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent reverser
}

waterglass_right
{
    mesh waterglass_right.im
    anim waterglass_right.kin
    auto-create 1
    limits 0, 100
    kind animated-dial
}

waterglass_left
{
    mesh waterglass_left.im
    anim waterglass_left.kin
    auto-create 1
    limits 0, 100
    kind animated-dial
}

firebox
{
    mesh firebox.im
    auto-create 1
    kind firebox

    light 0
    test-collisions 0
}

fire
{
    mesh fire.im
    auto-create 1
    light 0
    test-collisions 0
}

coal
{
    mesh coal.im
    auto-create 1
    light 0
    test-collisions 0
}

fireglow
{
    mesh fireglow.im
    auto-create 1
    light 0
    test-collisions 0
}

coalman
{
    mesh coalman/coalman.im
    auto-create 1
    att-parent default
    att a.coalman
    effects
    {
        shovel
        {
            kind animation
            anim coalman/Coalman_shovel.kin
        }
        wave
        {
            kind animation
            anim coalman/Coalman_wave.kin
        }
        idle1
        {
            kind animation
            anim coalman/Coalman_loop1.kin
        }
        idle2
        {
            kind animation
            anim coalman/Coalman_loop2.kin
        }
        loop2shovel
        {
            kind animation
            anim coalman/Coalman_loop2shovel.kin
        }
        shovel2loop
        {
            kind animation
            anim coalman/Coalman_shovel2loop.kin
        }
        wipebrow
        {
            kind animation
            anim coalman/Coalman_wipebrow.kin
        }
    }
}
}

```

PB15Cabin.gs

This is the PB15 steam interior script file.

This script is responsible for mapping all the steam-specific controls to the physics system, and for handling PB-15 specific functions such as the coal shovelling dude.

```
include "train.gs"
include "locomotive.gs"
include "cabin.gs"

class PB15CabinData isclass CabinData
{
    public float fire_plates_val;
    public float left_window_val;
    public float right_window_val;
    public float left_sliding_window_val;
    public float right_sliding_window_val;
    public float seat0_val;
    public float seat1_val;
    public float sanding_lever_val;
    public float whistle_lever_val;
    public float regulator_lever_val;
    public float blowdown_lever_val;
    public bool fireboxDoorOpen;
};

class PB15Cabin isclass Cabin
{
    Locomotive loco;

    CabinControl speedometer;
    CabinControl main_reservoir_needle;
    CabinControl brake_cylinder0_needle;
    CabinControl brake_cylinder1_needle;
    CabinControl no3_pipe_needle;
    CabinControl brake_pipe_needle;
    CabinControl equaliser_needle;
    CabinControl flow_needle;
    CabinControl ampmeter_needle0;
    CabinControl ampmeter_needle1;
    //CabinControl throttle_lever;
    CabinControl reverser_lever;
    CabinControl train_brake_lever;
    CabinControl train_lapbrake_lever;
    CabinControl loco_brake_lever;
    CabinControl dynamic_brake_lever;
    CabinControl wheelslip_light;
    CabinControl horn_rope;
    CabinControl pantograph_lever;
    CabinControl light_switch;

    CabinControl waterGlassLeft_dial, waterGlassRight_dial;
    CabinControl firebox;
    CabinControl boiler_needle;
    CabinControl regulator_lever;
    CabinControl fire_plates;
    CabinControl left_window;
    CabinControl right_window;
    CabinControl left_sliding_window;
    CabinControl right_sliding_window;
    CabinControl seat0;
    CabinControl seat1;
    CabinControl sanding_lever;
    CabinControl whistle_lever;
    CabinControl blowdown_lever;

    CabinControl waterInjector0, waterInjector1;

    PB15CabinData cabinData;
```



```

bool shovellingCoal;
bool waving;

thread void RunAnimation(void);

public void Init(void)
{
    speedometer = GetNamedControl("speedo_needle");
    main_reservoir_needle = GetNamedControl("bplocomain_needle");
    brake_cylinder0_needle = GetNamedControl("vbptrainbrakecylinder_needle");
    brake_cylinder1_needle = GetNamedControl("no3pipe_needle");
    no3_pipe_needle = GetNamedControl("bptrainbrakecylinder2_needle");
    brake_pipe_needle = GetNamedControl("bptrainbrakepipe_needle");
    equaliser_needle = GetNamedControl("vbploco_equaliser");
    flow_needle = GetNamedControl("flow_needle");
    ampmeter_needle0 = GetNamedControl("ampmeter_needle");
    ampmeter_needle1 = GetNamedControl("vampmeter2_needle");
    //throttle_lever = GetNamedControl("vthrottle_lever");
    train_brake_lever = GetNamedControl("trainbrake_lever");
    train_lapbrake_lever = GetNamedControl("trainbrakelap_lever");
    loco_brake_lever = GetNamedControl("vindependantbrake_lever");
    dynamic_brake_lever = GetNamedControl("dynamicbrake_lever");
    wheelslip_light = GetNamedControl("wheelslip_light");
    horn_ropo = GetNamedControl("horn");
    light_switch = GetNamedControl("light_switch");

    waterGlassLeft_dial = GetNamedControl("waterglass_left");
    waterGlassRight_dial = GetNamedControl("waterglass_right");

    firebox = GetNamedControl("firebox");
    boiler_needle = GetNamedControl("boiler_needle");
    regulator_lever = GetNamedControl("regulator");
    reverser_lever = GetNamedControl("reverser");

    waterInjector0 = GetNamedControl("vwater_injector_0");
    waterInjector1 = GetNamedControl("water_injector_1");

    fire_plates = GetNamedControl("fire_plates");
    left_window = GetNamedControl("left_window");
    right_window = GetNamedControl("right_window");
    left_sliding_window = GetNamedControl("left_sliding_window");
    right_sliding_window = GetNamedControl("right_sliding_window");
    seat0 = GetNamedControl("seat0");
    seat1 = GetNamedControl("seat1");
    sanding_lever = GetNamedControl("sanding_lever");
    whistle_lever = GetNamedControl("whistle_lever");
    blowdown_lever = GetNamedControl("blowdown_lever");

    RunAnimation();
}

//! Attach this cabin to a game object (i.e. a locomotive).
//
// Param:  obj  Game object to attach this cabin to (usually a Locomotive).
//
public void Attach(GameObject obj)
{
    loco = cast<Locomotive>(obj);

    // get cabin data
    CabinData cd = loco.GetCabinData();
    if(cd)
    {
        // reset the controls from saved values
        PB15CabinData pbcd = cast<PB15CabinData>cd;
        if(fire_plates)
            fire_plates.SetValue(pbcd.fire_plates_val);
        if(left_window)
            left_window.SetValue(pbcd.left_window_val);
        if(right_window)

```

```

        right_window.SetValue(pbcd.right_window_val);
        if(left_sliding_window)
            left_sliding_window.SetValue(pbcd.left_sliding_window_val);
        if(right_sliding_window)
            right_sliding_window.SetValue(pbcd.right_sliding_window_val);
        if(seat0)
            seat0.SetValue(pbcd.seat0_val);
        if(seat1)
            seat1.SetValue(pbcd.seat1_val);
        if(sanding_lever)
            sanding_lever.SetValue(pbcd.sanding_lever_val);
        if(whistle_lever)
            whistle_lever.SetValue(pbcd.whistle_lever_val);
        if(regulator_lever)
            regulator_lever.SetValue(pbcd.regulator_lever_val);
        if(blowdown_lever)
            blowdown_lever.SetValue(pbcd.blowdown_lever_val);
    }
    else
    {
        PB15CabinData pbd = new PB15CabinData();
        loco.SetCabinData(pbd);
    }
}

// the default locomotive config uses kPa to describe pressure dial ranges
// this function converts from g/m^3 into kPa-above-atmospheric
public float GetPressureParam(string param)
{
    float pressureMultiplier = 1.0 / (0.145 * 0.0000703);
    float pressureBase = 14.7 * 0.0000703;

    return pressureMultiplier * (loco.GetEngineParam(param) - pressureBase);
}

public void Update(void)
{
    float value;
    Train train = loco.GetMyTrain();

    //
    // Update Needles
    //

    if (speedometer)
        speedometer.SetValue(train.GetVelocity());

    if (main_reservoir_needle)
        main_reservoir_needle.SetValue(GetPressureParam("main-reservoir-pressure"));

    value = GetPressureParam("brake-cylinder-pressure");
    if (brake_cylinder0_needle)
        brake_cylinder0_needle.SetValue(value);
    if (brake_cylinder1_needle)
        brake_cylinder1_needle.SetValue(value);

    if (no3_pipe_needle)
        no3_pipe_needle.SetValue(GetPressureParam("no3-pipe-pressure"));

    if (brake_pipe_needle)
        brake_pipe_needle.SetValue(GetPressureParam("brake-pipe-pressure"));

    if (equaliser_needle)
        equaliser_needle.SetValue(GetPressureParam("equaliser-pressure"));

    if (flow_needle)
        flow_needle.SetValue(GetPressureParam("flow"));

    value = loco.GetEngineParam("current-drawn");
    if (ampmeter_needle0)

```

```

ampmeter_needle0.SetValue(value);
if (ampmeter_needle1)
ampmeter_needle1.SetValue(value);

//
// Update Levers
//
// This is done in case the game logic has changed any of the settings from what
// the user selected.
//

//if (throttle_lever)
// throttle_lever.SetValue(loco.GetEngineSetting("throttle")); // * 8.0);

if (reverser_lever)
reverser_lever.SetValue(loco.GetEngineSetting("reverser"));

if (train_brake_lever)
train_brake_lever.SetValue(loco.GetEngineSetting("train-auto-brake"));

if (train_lapbrake_lever)
train_lapbrake_lever.SetValue(loco.GetEngineSetting("train-lap-brake"));

if (loco_brake_lever)
loco_brake_lever.SetValue(loco.GetEngineSetting("loco-auto-brake"));

if (dynamic_brake_lever)
dynamic_brake_lever.SetValue(loco.GetEngineSetting("dynamic-brake"));

if (wheelslip_light)
wheelslip_light.SetValue(loco.GetEngineParam("wheelslip"));

if (horn_rope)
horn_rope.SetValue(loco.GetEngineParam("horn"));

if (pantograph_lever)
pantograph_lever.SetValue(loco.GetEngineSetting("pantograph"));

if (light_switch)
light_switch.SetValue(loco.GetEngineSetting("headlight"));

if (waterGlassLeft_dial)
waterGlassLeft_dial.SetValue(loco.GetEngineParam("steam-boiler-liquid-percent"));

if (waterGlassRight_dial)
waterGlassRight_dial.SetValue(loco.GetEngineParam("steam-boiler-liquid-percent"));

// update cabin data
PB15CabinData cd = cast<PB15CabinData>loco.GetCabinData();
if(left_window)
cd.left_window_val = left_window.GetValue();
if(right_window)
cd.right_window_val = right_window.GetValue();
if(left_sliding_window)
cd.left_sliding_window_val = left_sliding_window.GetValue();
if(right_sliding_window)
cd.right_sliding_window_val = right_sliding_window.GetValue();
if(seat0)
cd.seat0_val = seat0.GetValue();
if(seat1)
cd.seat1_val = seat1.GetValue();
if(sanding_lever)
cd.sanding_lever_val = sanding_lever.GetValue();
if(whistle_lever)
cd.whistle_lever_val = whistle_lever.GetValue();
if(regulator_lever)
cd.regulator_lever_val = regulator_lever.GetValue();
if(fire_plates)
cd.fire_plates_val = fire_plates.GetValue();
if(blowdown_lever)

```

```

cd.blowdown_lever_val = blowdown_lever.GetValue();

cd.fireboxDoorOpen = fire_plates.GetValue() > 0.9;

if (firebox)
{
    firebox.SetNamedValue("amount-burning-coal", 0.5);

    if (fire_plates)
    {
        firebox.SetNamedValue("door-open", fire_plates.GetValue());
    }

    firebox.SetNamedValue("fire-life", loco.GetEngineParam("fire-temperature") / 1600.0);
    firebox.SetNamedValue("steam-piston-cycle", loco.GetEngineParam("steam-piston-cycle"));
}

if (boiler_needle)
{
    boiler_needle.SetValue(GetPressureParam("steam-boiler-pressure"));
}

if (waterInjector0)
waterInjector0.SetValue(loco.GetEngineSetting("injector"));

if (waterInjector1)
waterInjector1.SetValue(loco.GetEngineSetting("injector"));
}

void UserSetControl(CabinControl p_control, float p_value)
{
    Interface.Log("control: " + p_control.GetName() + " value: " + p_value);

    if (p_control == reverser_lever)
        loco.SetEngineSetting("reverser", p_value);

    // else if (p_control == throttle_lever)
    // loco.SetEngineSetting("throttle", p_value);

    else if (p_control == train_brake_lever)
        loco.SetEngineSetting("train-auto-brake", p_value);

    else if (p_control == train_lapbrake_lever)
        loco.SetEngineSetting("train-lap-brake", p_value);

    else if (p_control == loco_brake_lever)
        loco.SetEngineSetting("loco-auto-brake", p_value);

    else if (p_control == dynamic_brake_lever)
        loco.SetEngineSetting("dynamic-brake", p_value);

    else if (p_control == horn_rope)
        ; // todo

    else if (p_control == pantograph_lever)
        loco.SetEngineSetting("pantograph", p_value);

    else if (p_control == light_switch)
        loco.SetEngineSetting("headlight", p_value);

    else if (p_control == regulator_lever)
        loco.SetEngineSetting("regulator", p_value);

    else if (p_control == waterInjector0 or p_control == waterInjector1)
        loco.SetEngineSetting("injectorv", p_value);
}

void UserPressKey(string s)
{
    if (s == "shovel-coal")

```

```

{
    PB15CabinData cd = cast<PB15CabinData>loco.GetCabinData();
    // if(cd.fireboxDoorOpen)
    {
        if(FireAddCoal())
        {
            if(!shovellingCoal)
            {
                if(!waving)
                {
                    if(fire_plates)
                        fire_plates.SetValue(1.0f);

                    shovellingCoal = true;
                    SetFXAnimationState("idle1", false);
                    SetFXAnimationState("idle2", false);
                    SetFXAnimationState("loop2shovel", false);
                    SetFXAnimationState(vloop2shovel", true);
                }
            }
        }
    }
}
else if(s == "wave")
{
    if(!shovellingCoal)
    {
        if(!waving)
        {
            waving = true;
            SetFXAnimationState("wave", false);
            SetFXAnimationState("wave", true);
        }
    }
}
}

thread void RunAnimation(void)
{
    SetFXAnimationState("idle1", true);

    wait()
    {
        on "Animation-Event", "Coalman_loop1_end":
            SetFXAnimationState("idle1", false);
            SetFXAnimationState("idle2", true);
            continue;

        on "Animation-Event", "Coalman_loop2_end":
            SetFXAnimationState("idle2", false);
            SetFXAnimationState("idle1", true);
            continue;

        on "Animation-Event", "Coalman_loop2shovel_end":
            SetFXAnimationState("loop2shovel", false);
            SetFXAnimationState("shovel", true);
            continue;

        on "Animation-Event", "Coalman_shovel_end":
            SetFXAnimationState("shovel", false);
            SetFXAnimationState("wipebrow", true);
            continue;

        on "Animation-Event", "Coalman_wipebrow_end":
            SetFXAnimationState("wipebrow", false);
            SetFXAnimationState("shovel2loop", true);
            continue;

        on "Animation-Event", "Coalman_shovel2loop_end":
            SetFXAnimationState("shovel2loop", false);

```

```
SetFXAnimationState("idle1", true);
shovellingCoal = false;
continue;

on "Animation-Event", "Coalman_wave_end":
SetFXAnimationState("wave", false);
SetFXAnimationState("idle1", true);
waving = false;
continue;

}

}

};
```

KIND: PANTOGRAPH

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\pants

These are the animated mechanisms on the roof of electric locomotives that conduct to an electric catenary (wires) above.

Referenced by the *pantograph* tag in a traincar config.txt

Config.txt:

```
kuid <KUID2:###:#####:1>
kind pantograph
obsolete-table
{
}
username
description " "
trainz-build 1.5
category-class AL
category-region-0 US
category-era-0 1960s
```

KIND: ENVIRONMENT

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Environment\

Config.txt:

```
kuid <KUID2:###:#####:1>
kind environment
region Britain
normal mediumclouds
storm mediumClouds_Storm
night mediumClouds_Night
obsolete-table {
}
username
description " "
trainz-build 1.5
category-class ES
category-region-0 UK
category-era-0
```

Breakdown for Environment Config.txt:

region

Surveyor region.

normal

Name of image file for normal sky.

File should be 256 x 256 pixel 24bit tga.

storm

Name of image file for stormy sky.

File should be 256 x 256 pixel 24bit tga.

night

Name of image file for night sky.

File should be 256 x 256 pixel 24bit tga.

KIND: WATER2

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Environment\

Note: New animated water in TRS (kind water2) supersedes previous UTC water (kind water).

Therefore kind water is no longer supported by TRS.

Where a UTC map refers to an old kind water asset it will automatically be updated to one of the new animated water assets.

Bump-mapping requires special settings and naming conventions in 3dsmax. Refer to [Page 99](#) and the following download for information:

http://www.auran.com/TRS2004/downloads/contentcreation/TRS_Max4_Plugin_Bump.zip

Essentially the new water system comprises of an animated, bump-mapped, transparent surface. The animation variables are controlled through the water.anim.txt. The tile size and material properties are controlled through the water.config.txt file (following page).

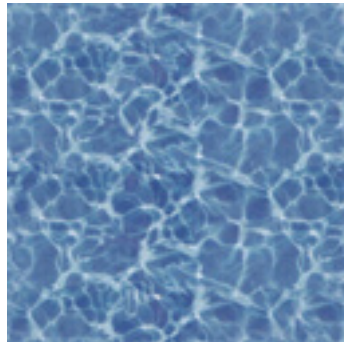
The following example is of Calm Water. Choppy, Glassy and Rough water are described on the following pages.

Calm Water

config.txt *Calm Water*

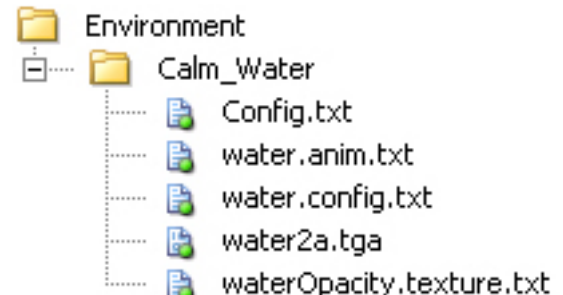
```
kuid <KUID2:###:#####:1>
kind water2
username "Glassy water"
obsolete-table {
}
description " "
trainz-build 2.4
category-class EW
category-region-0 UK
category-era-0
```

water2a.tga



256 x 256 pixel 32bit tga.

Typical Water2 Directory Structure



water.anim.txt *Calm Water* *Note: // indicates start of creator notes only (comments on script)*

```
version 1.00

// Water DetailAnim configuration file
// Is used from DefaultWater.config.txt
DetailAnim
{
    animSampleRate = 10;           // Sample rate (samples per sec)
    animPeriod = 5;                // Looping period in sec.
    animSpeed = 2.0;               // Speed of waves
    animSize = 128,128;            // Bump map dimentions
    animWorldSize = 450.0;         // "Size" of one tile
    animMaxHeight = 0.4;           // Max height of the wave
    animScaleNormXY = 4.0;         // scale X,Y coordinates of the normal map for better interpolation

    FFT
    {
        animFFTWindVec = -15.0,5.0; // Direction and speed of the wind affecting length of the waves
        animPhillipsA = 1.0e-3;     // Phillips spectrum constant affecting heights of the waves
        animFFTSeed = 0;
    }
}
```

water.config.txt *Calm Water Note: // indicates start of creator notes only*

```
version 1.00

// WaterManager config data
WaterManager("WaterManagerGeneric")
{
    WaterMaterial
    {
        materialColor = (0.20, 0.45, 0.45, 0.8);
        materialRI = 0.3;
        opacityTex = WaterOpacity.texture;
        opacityAmount = 0.5;
    }

    // Compiled DetailAnim or text ConfigData file 'water.anim.txt'
    // This is now loaded manually by Trainz so Trainz can cache the anim file in a separate folder.
    // DetailAnimFile = water.anim;
}

// WaterGeometry config data
WaterGeometry
{
    UVScrollVelocity = 0.0, 0.05;
    TileUVScale = 1.0, 1.0;

    GridSpacing = 10.0; // "Size" of one cell of the grid (is used if MaxAmp > 0)
    TileGridSize = 2, 2; // Number of vertices in one tile (use more if MaxAmp > 0)
    WaveFreq = 0.0; //0.15;
    MaxAmp = 0.0; //0.25;

    // Mesh animation
    // TileGridSize = 3, 3; // Number of vertices in one tile (use more if MaxAmp > 0)
    // WaveFreq = 0.15;
    // MaxAmp = 0.25;
}
```

Choppy Water

Notation (comments on the script) removed for clarity.

config.txt *Calm Water*

```
kuid <KUID2:###:#####:1>
kind water2
username "Choppy water"
obsolete-table {
}
description " "
trainz-build 2.0
category-class EW
category-region-0 UK
category-era-0
```

water.anim.txt *Choppy Water*

```
version 1.00

DetailAnim
{
    animSampleRate = 10;
    animPeriod = 5;
    animSpeed = 5.0;
    animSize = 128,128;
    animWorldSize = 450.0;
    animMaxHeight = 1.0;
    animScaleNormXY = 4.0;

    FFT
    {
        animFFTWindVec = -15.0,5.0;
        animPhillipsA = 1.0e-3;
        animFFTSeed = 0;
    }
}
```

water.config.txt *Choppy Water*

```
version 1.00

WaterManager("WaterManagerGeneric")
{
    WaterMaterial
    {
        materialColor = (0.20, 0.45, 0.45, 0.8);
        materialRI = 0.3;
        opacityTex = WaterOpacity.texture;
        opacityAmount = 0.5;
    }
}

WaterGeometry
{
    UVScrollVelocity = 0.0, 0.05;
    TileUVScale = 1.0, 1.0;
    GridSpacing = 10.0;
    TileGridSize = 2, 2;
    WaveFreq = 0.0;
    MaxAmp = 0.0;
}
```

Glassy Water

Notation (comments on the script) removed for clarity

config.txt *Glassy Water*

```
kuid <KUID2:###:####:1>
kind water2
username "Glassy water"
obsolete-table {
}
description " "
trainz-build 2.4
category-class EW
category-region-0 UK
category-era-0
```

water.anim.txt *Glassy Water*

```
version 1.00

DetailAnim
{
    animSampleRate = 1;
    animPeriod = 1;
    animSpeed = 1.0;
    animSize = 128,128;
    animWorldSize = 450.0;
    animMaxHeight = 0.0;
    animScaleNormXY = 4.0;

    FFT
    {
        animFFTWindVec = 0.0,0.0;
        animPhillipsA = 1.0e-3;
        animFFTSeed = 0;
    }
}
```

water.config.txt *Glassy Water*

```
version 1.00

WaterManager("WaterManagerGeneric")
{
    WaterMaterial
    {
        materialColor = (0.20, 0.45, 0.45, 0.8);
        materialRI = 0.3;
        opacityTex = WaterOpacity.texture;
        opacityAmount = 0.5;
    }
}

WaterGeometry
{
    UVScrollVelocity = 0.0, 0.05;
    TileUVScale = 1.0, 1.0;
    GridSpacing = 10.0;
    TileGridSize = 2, 2;
    WaveFreq = 0.0;
    MaxAmp = 0.0;
}
```

Rough Water

Notation (comments on the script) removed for clarity

config.txt *Rough Water*

```
kuid <KUID2:###:####:1>
kind water2
username "Rough water"
obsolete-table {
}
description " "
trainz-build 2.4
category-class EW
category-region-0 UK
category-era-0
```

water.anim.txt *Rough Water*

```
version 1.00

DetailAnim
{
    animSampleRate = 10;
    animPeriod = 5;
    animSpeed = 5.0;
    animSize = 128,128;
    animWorldSize = 450.0;
    animMaxHeight = 2.0;
    animScaleNormXY = 4.0;

    FFT
    {
        animFFTWindVec = -12.0,4.0;
        animPhillipsA = 1.0e-3;
        animFFTSeed = 0;
    }
}
```

water.config.txt *Rough Water*

```
version 1.00

WaterManager("WaterManagerGeneric")
{
    WaterMaterial
    {
        materialColor = (0.20, 0.45, 0.45, 0.8);
        materialRI = 0.3;
        opacityTex = WaterOpacity.texture;
        opacityAmount = 0.5;
    }
}

WaterGeometry
{
    UVScrollVelocity = 0.0, 0.05;
    TileUVScale = 1.0, 1.0;
    GridSpacing = 10.0;
    TileGridSize = 2, 2;
    WaveFreq = 0.0;
    MaxAmp = 0.0;
}
```

KIND: MAP

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\maps\

The config.txt for maps are automatically generated by TRS Surveyor.

You can add a soundscript to the config if desired such as the example below. Refer Soundscripts [Page 126](#).

.

Config.txt :

```
kind map
kuid <KUID2:###:#####:1>
username Britain
workingscale 0
workingunits 0
water <KUID:-1:8009>
region Britain
trainz-build 2.4
soundscript
{
  morning
  {
    ambient 1
    value-range 1, 0.1
    volume 0.3
    sound
    {
      ctry_day_1.wav
    }
  }
  night
  {
    ambient 1
    value-range 0, 0.9
    volume 0.3
    sound
    {
      night_loop.wav
    }
  }
}
....
```

KIND: PROFILE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\profiles\

A Profile is otherwise known as a Session in TRS.

You can now create multiple Sessions in Surveyor for a single route with different consists, starting points, industry outputs etc. You may have different sets of trains in each different session.

The config.txt and profile.dat files for Profiles are automatically generated by TRS Surveyor and generally shouldn't be edited.

KIND: GROUNDTEXTURE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Ground\

These are the text.

Config.txt :

```
kuid <KUID2:###:#####:1>
kind groundtexture
region Britain
obsolete-table
{
}
clutter-mesh <KUID2:###:#####:1>

username
description " "
trainz-build 2.4
category-class GL
category-region-0 UK
category-era-0
```

Breakdown of Groundtexture Config.txt:

region

Surveyor region.

clutter-mesh

Not a requirement. Optional only. See below.

GROUNDTEXTURE CLUTTER MESH

Ground textures can now reference a mesh and insert the mesh automatically as the ground is painted. Painting over a clutter-mesh ground texture effectively deletes the clutter meshes and texture.

The mesh it refers to is can be *standard scenery object* or *kind mesh*.

Clutter-meshes must have only one 3dsmax material assigned to it only. Polycounts must be very low.

Note: The draw distance is very short for clutter-mesh assets. We suggest only scrubs or grasses be used like the example below.

Ground Clutter Mesh



KIND: SCENERY

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenery\

The example right shows how flexible the mesh table can be. Note this asset has 2 animations. One as part of the default mesh, and another external mesh inserted into an attachment point within the default mesh (the radar).

Note: We no longer need the UTC tag `autoanimation 1` as all animations are now referenced through the mesh-table.

The night mesh (default-night) has also changed from UTC. It too is now referenced through the mesh-table.

Breakdown of Scenery Config.txt:

Please refer to earlier config.txt examples for other entries not described here.

region

Surveyor region.

type

Surveyor type.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is lit by general light value (uniform colouring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

nightmode

Only add this tag if you reference a default-night mesh in the mesh-table.

Values: home, lamp or constant.

Home switches on night effect at dusk and off sometime during the night. Lamp switches the night effect on from dusk to dawn. Constant lights are on day and night.

corona and name effects (optional)

Refer Effects - [Page 13](#)

SCENERY MOVEMENT VARIABLES

snapgrid n

Where n is a value in meters. This lets you specify the size of the grid the object snaps to.

We recommend factors/fractions of 720 as this is the size of a base board and the positioning may do funny things across section borders. e.g. 1, 2, 5, 10, 20, 30, 40, 45, 60, 80, 90, 120, 180, 240, 360, 720.

The default snapgrid is 10. See snapmode (below) on how to enable grid snapping.

rotstep n

Where n is a value in degrees. This lets you specify the

Typical Scenery Config.txt :

```
kuid <KUID2:###:####:1>
kind scenery
region Britain
type Buildings
light 1
nightmode home
username
kuid-table{
}
obsolete-table {
}
mesh-table
{
  default
  {
    mesh scenery_asset.im
    anim anim.kin
    auto-create 1
    animation-loop-speed 1
    effects
    {
      0
      {
        kind name
        fontsize 0.15
        fontcolor 30,30,30
        att a.name0
        name name
      }
      1
      {
        kind corona
        att a.coronawhite
        frequency 1
        directional 0
        texture-kuid <KUID:-3:10111>
      }
    }
  }
  default-night
  {
    mesh night.im
    night-mesh-base default
  }
  radar
  {
    mesh radar/radar.im
    anim radar/radar.kin
    att a.radar
    att-parent default
    animation-loop-speed 1.0
  }
}

description " "
trainz-build 2.4
category-class FS
category-region-0 UK
category-era-0
```

*step size of rotation angles for this object. Other example values are 1, 10, 20, 90, 180 etc.
The default rotstep is 1.0*

snapmode n

Where n is either 0 (default) , 1 or 2. Use snapmode to enable snapping of a scenery object to the snap grid.

*0 will disable grid snapping (default)
1 will enable grid snapping
2 will enable an offset grid snapping.*

Offset grid snapping will cause objects to be snapped to the grid but will also offset the object's position by ½ the grid size – essentially positioning the object in between the normal grid lines.

See snapgrid (above) on how to set the snap grid size from the default of 10 meters.

Note: In TRS, you can now hold down the control key while moving object's to snap them to the default grid of 10 meters.

rotate n

Where n is 0 or 1 (default). This lets you disable rotation on a scenery object.

*0 to disable
1 to enable (default).*

rotate-yz-range min, max

eg: rotate-yz-range -90, 90

Where min and max are values in degrees. This tag lets you set the roll / yz rotation range (normal object rotation is an xy rotation). If you want your scenery object to support rolling then use this tag to set the minimum and maximum roll range. By default, objects have a min/max roll range of 0 to 0.

rollstep n

Where n is a value in degrees. Used in conjunction with rotate-yz-range, rollstep lets you specify the step size of roll angles for this object. Other example values are 1, 5, 20 etc. The default rollstep is 1.0.

height-range min, max

eg: height-range -10, 100

Where min and max are values in meters.

Custom scenery objects are height adjustable, and this tag allows you to specify the minimum and maximum height ranges for changing the height with the new "Adjust Height" tool in Surveyor's 'Object Tools' panel.

Adding a height range is particularly useful for ships/ buoys (placed on water) and for Station accessories.

Note: Animation Events

Sounds events and generic events can be linked to an animation key-frame to give great control over sound and script timing for industry and scenery assets. Refer to [Page 104](#).

KIND: INDUSTRY

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenery\

An industry is best described as a scenery asset with product processing functionality. Industry assets interact with compatible rolling stock assets through their script file. *See note at bottom of page.*

Tags that apply in a scenery asset may also apply in an industry asset. See Kind Scenery [Page 72](#).

Full config or script files are not included in this section because all TRS released industry config.txt files and script files (.gs) are available for download reference:

<http://www.auran.com/TRS2004/trssp4dl/dpage.php>

Overview

All TRS industries have attached-tracks, attached-triggers, queues and processes to input or output products (or commodities).

Attached-track (required)

Attached tracks are setup in an industries config.txt. Auto-generated spline track is generated through attachment points located within the default mesh.

*Attached-tracks update automatically to the spline track connected to it. You may over-ride this auto-update feature by adding **useadjoiningtracktype 0***

Note. Correct track end attachment orientation is essential. The Y axis must point 'out' at the correct angle. the Z axis must point 'up'. Refer to [Page 82](#).

```
attached-track
{
    track0
    {
        track <KUID:-3:15>
        useadjoiningtracktype 0
        vertices
        {
            0 a.track0a
            1 a.track0b
        }
    }
}
```

Attached-trigger (required)

A Trigger is a point along an attached track with a specified radius. When a compatible rolling stock item enters this radius it triggers a set of commands, controlled through its script.

Note: The increasing use of scripts in TRS adds huge flexibility and control to assets and their functionality.

Adding scripts on a per asset basis is a logical progression in the development of Trainz. However, we do understand that most 3D modelers do not know a great deal about scripts let alone how to write them. With the release of TRS, we are really dawning on a new era in Trainz custom content creation.

Because of this, we recognise there will need to be far more collaboration and group efforts for custom industry asset creation. There are several very good script writers in the Trainz content Creation community. Just ask around the forums.

A trigger is setup in an industries config.txt.

Queues (required)

The queues field states which product or products the industry can use. It contains the size of each product, the initial count when placed, and can refer to it's visual load state whether through a load animation or attachment. Any load animations are set-up within the mesh-table.

Processes (required)

The input and output settings of the industry.

You can specify the amount of input and output for each queue referenced product as well as the duration (or rate) in seconds for that process to take place.

All queues and processes are linked through the industry asset's script file.

Industry functionality

Perhaps the simplest examples of industry functionality are the TRS released Coalmine and the Powerstation assets.

When the coal hopper enters the trigger radius of the coalmine loading bay, it's script interacts with the hoppers own script. Particle effects (pfx) from the coalmine visually display the coal entering the hopper and the hopper animated load rises to show it's full state. The coalmine's own animated load pile reduces as does it's commodity level.

Similarly, when the full hopper enters the Powerstation trigger radius, the hopper's animated load lowers, the side doors open and the pfx effects on the hopper itself initiate. The animated load pile in the Powerstation increases and it's commodity level increases.

The hopper pfx, and the animated doors are both controlled by the hopper.gs script file.

Refer: IN-GAME VISUALISATION OF PRODUCTS [Page 18](#).

Note: Animation Events

Sounds events and generic events can be linked to an animation key-frame to give great control over sound and script timing for industry and scenery assets. Refer to [Page 104](#).

Note: Preview Window Icons

Auran released Industry assets (and industry compatible assets) have an additional tag in their config: icon0 <KUID:-3:10164>

The kuid is of a kind texture and the icon texture is a 32x32 pixel - 32 bit tga file and you may have up to 4 (icon0, icon1, icon2, icon3)

PORTAL

A portal is an industry object used to create and emit trains. It relies on a script to function. An example config.txt file is shown for a basic portal asset.

HTML lines have been omitted under the string-table. The working asset has extensive entries here to make it fully functional. Refer to the example Portal asset available on the Auran Content Creation Art Source CDs.

Breakdown of Portal Config.txt:

Some config.txt tags are not explained below. Others are covered in the general config.txt explanation, see [Page 12](#).

region

Surveyor region.

type

Surveyor type.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is lit by general light value (uniform coloring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

script

Name of Script to be used with this asset.

class

Script class.

icon-texture

The in-game representation of the asset when specifying a drive to command in Driver.

category-class

The class code for this asset. Refer to [Page 5](#)

kuid-table

KUIDs required for this asset to function correctly, sample only shown (the rolling stock listed in the consists entry).

default

Default is the 'main' mesh of the asset.

mesh

The model name.

auto-create 1

The model is generated automatically when placed, or when you load a map which includes the model. In some instances you don't want the mesh visible (as this may be controlled through script). If auto-create is 0 the mesh will not be visible when placed.

Portal Config.txt:

```
kuid <KUID2:###:#####:1>
kind industry
region Australia
type Industrial
light 1
kuid-table {
}
obsolete-table {
}
description " "
trainz-build 2.4
username Portal
script PortalTunnel
class PortalTunnel
icon-texture icon_portal.tga
category-region-0 AU
category-era-0 2000s
category-class BIN

kuid-table
{
  loco <kuid:-1:100861>
  wagon <kuid:-1:100048>
  guardsvan <kuid:-1:100770>
}

mesh-table
{
  default
  {
    mesh portal.im
    auto-create 1
    effects
    {
      portalexentry
      {
        kind attachment
        att a.track0g
        default-mesh <KUID:-3:10239>
        surveyor-only 1
      }
      portalexend
      {
        kind attachment
        att a.track0a
        default-mesh <KUID:-3:10238>
        surveyor-only 1
      }
    }
  }
}

string-table
{
}

consists
{
  aus-freight
  {
    0
    {
      vehicle <kuid:-1:100725>
      running-number 4840
    }
    1
    {
      vehicle <kuid:-1:100828>
    }
    2
    {
      vehicle <kuid:-1:100828>
    }
  }
}
```

effects

kind attachment

Attaching a mesh to the default mesh using the kind attachment effect. Each effect is given a name such as "portalentry".

default-mesh

Kuid of the mesh to be attached.

att

The mesh is inserted at a mesh attachment point rather than the origin (without this line the mesh is placed relative to the origin of the parent model).

surveyor-only 1

The attached mesh will only be visible in Surveyor.

string-table

A listing of HTML options to be called by the script file to display when the properties option of the asset is used in Surveyor. H lines have been omitted in this example. Refer to the example Portal model available on the Auran Content Creation Art Source CDs.

consists

List of the kuid numbers of the cars to make up a consist for production by the portal.

running-number

The default number for the vehicle. It is changeable in Surveyor.

facing

Determines the vehicle facing direction.
0 represents reversed.
1 represents forward (default).

show-in-consist-menu

Indicates if the created train should be user-controllable.
0 reserves the train as a script-controlled train.
1 causes the train to be user-controllable.

coupling-mask

Indicates if the created train can be coupled to.
0 = the train can not be coupled to.
1 = the train can be coupled to.

decoupling-mask

Indicates if the created train is permitted to be decoupled.
0 = the train can not be decoupled.
1 = the train can be decoupled.

attached-track

Track to be attached to the model, including a name for the track (out_track0), the track KUID to be used, and attachment points, placed in the 3dsmax/gmax model.

attached-trigger

A point on the attached track with a specified radius. When a compatible rolling stock item enters this radius it triggers a set of commands, controlled through script.

icon0

Window preview icon - see information box [Page 75](#).

```
3
{
    vehicle <kuid:-1:100048>
}
4
{
    vehicle <kuid:-1:100048>
}
vehicle <kuid:-1:100770>
}
show-in-consist-menu 0
coupling-mask 0
decoupling-mask 0
}
british-hst
{
    0
    {
        vehicle <kuid:-1:100039>
        running-number 7
    }
    1
    {
        vehicle <kuid:-1:100058>
    }
    2
    {
        vehicle <kuid:-1:100059>
        facing 0
    }
    3
    {
        vehicle <kuid:-1:100062>
        facing 0
    }
    show-in-consist-menu 0
    coupling-mask 0
    decoupling-mask 0
}
attached-track
{
    out_track0
    {
        track <KUID:-1:15>
        vertices
        {
            0 a.track0a
            1 a.track0b
            2 a.track0c
            3 a.track0d
            4 a.track0e
            5 a.track0f
            6 a.track0g
        }
    }
}
attached-trigger
{
    trig_end
    {
        att a.track0d
        radius 10.0
    }
    trig_entry
    {
        att a.track0f
        radius 10.0
    }
}
icon0 <kuid:-3:10164
```

MULTI INDUSTRY NEW

This is an industry object used to set up your own industry, with loading and unloading of products on attached track.

The example model consists of track with attachment points and triggers, allowing you to place your own buildings to suit. It relies on a script to function. An example config.txt file is shown for the asset.

Breakdown of Multi Industry New Config.txt:

Some config.txt tags are not explained below. Others are covered in the general config.txt explanation, see [Page 12](#).

script

Name of Script to be used with this asset.

class

Script class.

icon-texture

The in-game representation of the asset when specifying a "Drive To" command in Driver.

preview-mesh-kuid

As spline tracks will not render in the Preview window a preview-mesh is needed, as a kind mesh (this example model consists of spline track only).

kuid-table

A list of KUIDs required for this asset to function correctly, in this case the products to be supported.

default

Default is the 'main' mesh of the asset.

mesh

The 'main' mesh name.

auto-create 1

The model is generated automatically when placed, or when you load a map which includes the model. In some instances you don't want the mesh visible (as this may be controlled through script). If auto-create is 0 the mesh will not be visible when placed.

effects

The effects to be attached to the model.

arrow0

The name of the effect, in this case a red arrow to be attached to the model to shows the ends of the track.

att

The mesh is inserted at a mesh attachment point rather than the origin (without this line the mesh is placed relative to the origin of the parent model).

default-mesh

The mesh for the arrow model to be attached.

surveyor-only 1

The attached mesh will only be visible in Surveyor.

Multi Industry New Config.txt

```
kuid <KUID2:###:####:1>
kind industry
region Australia
type Industrial
light 1
kuid-table {
}
description " "
trainz-build 2.4
username Multiple_Industry_New
obsolete-table {
0 <kuid:-19:10152>
}
nightmode lamp
script MultipleIndustryNew
class MultipleIndustryNew
icon-texture icon_multiple.tga
preview-mesh-kuid <kuid:-3:10154>
kuid-table {
    coal <kuid:44179:60013>
    diesel <kuid:-3:10011>
    cont20ft <kuid:-3:10014>
    cont40ft <kuid:-3:10041>
    gengoods <kuid:-3:10013>
    logs <kuid:-3:10001>
    lumber <kuid:-3:10003>
    woodchips <kuid:-3:10002>
    oil <kuid:-3:10010>
    petrol <kuid:-3:10012>
    water <kuid:-3:10004>
    avgas <kuid:-3:10045>
}
mesh-table {
    default {
        mesh Multiple_Industry.im
        auto-create 1
        effects {
            arrow0 {
                att a.track0a
                default-mesh <kuid:-3:10092>
                surveyor-only 1
                kind attachment
            }
            arrow1 {
                att a.track0f
                default-mesh <kuid:-3:10092>
                surveyor-only 1
                kind attachment
            }
        }
    }
}
attached-track {
    out_track0 {
        track <kuid:-1:15>
        vertices {
            0 a.track0a
            1 a.track0b
            2 a.track0c
            3 a.track0d
            4 a.track0e
            5 a.track0f
        }
    }
}
attached-trigger {
    trig0 {
        att a.track0b
        radius 10
    }
    trig1 {
        att a.track0c
        radius 10
    }
}
```

kind

Kind attachment.

attached-track

Details of the track to be attached to the model, defined by the attachment points in the 3dsmax/gmax model using the a.name convention. Note the axis orientation of the end attachment points. Refer to the Fixed Track example [Page 82](#).

out_track0

Name of the track.

track

The track type (kuid) to be used for the attachment, in this case the red invisible track kuid.

vertices

List of attachment vertices from the 3dsmax/gmax model, for track attachment.

attached-trigger

A point inserted in 3dsmax/gmax, on the attached track, with a specified radius in the config.txt file. When a compatible rolling stock item enters this radius it triggers a set of commands, controlled through script.

trig0

Name of the trigger.

radius

The radius of operation in metres.

queues

The queues field defines the queue name, the product, the size and the initial count when placed.

size

The initial size of the queue (in product units)

allowed-products

The allowed products, not specified, therefore allows multiproducts listed in the kuid-table.

processes

The input and output settings of the passenger asset. You can specify the amount of input and output for each queue referenced product as well as the duration (or rate) in seconds for that process to take place.

start-enabled 1

Option is set such that this process will be running by default when the session is launched

duration

Duration time of the process in seconds.

inputs

Input quantity (in product units) for the process.

outputs

Output quantity (in product units) for the process.

icon0

Window preview icon - see information box [Page 74](#).

```

    }
    trig2 {
        att    a.track0d
        radius 10
    }
    trig3 {
        att    a.track0e
        radius 10
    }
}
queues {
    queue1 {
        size    100
        allowed-products {
        }
    }
    queue2 {
        size    100
        allowed-products {
        }
    }
    queue3 {
        size    100
        allowed-products {
        }
    }
    queue4 {
        size    100
        allowed-products {
        }
    }
    queue5 {
        size    100
        allowed-products {
        }
    }
    queue6 {
        size    100
        allowed-products {
        }
    }
}
processes {
    multi_consumer_producer {
        start-enabled 1
        duration      30
        inputs {
        }
        outputs {
        }
    }
}
icon0      <kuid:-3:10164>

```

PASSENGER STATION ASSET

A passenger station industry asset has more attributes than normal industry assets. It allows loading and unloading of passengers, and a "spawn" and "delete" process for passengers. It relies on a script to function.

An example config.txt file is shown for the a basic two platform passenger station asset. Some similar multiple attachment point entries in the original asset have been omitted for brevity.

For more information refer to the Passenger Station and Vehicle Assets Tutorial available for download:
http://files.auran.com/TRS2004/downloads/contentcreation/SP2-Passenger_Asset_Tutorial.zip

Breakdown of Passenger Station Config.txt:

Some config.txt tags are not explained below. Others are covered in the general config.txt explanation, see [Page 12](#).

icon-texture

The icon for the asset, used for the "Drive To" command, the file is a 64x64 tga texture with no alpha channel.

passenger-height

This value sets the height of the passenger asset in metres, to suit the platform model height.

queues

The queues field defines the passenger product, the size and the initial count when placed. Passenger attachment points placed in 3dsmax/gmax are referenced, but only a limited number have been included in this example.

passenger_on_X

Queue name for the passenger on platform, and must be of this form, where "X" is the platform number, starting at 0. Passenger off queues must be named similarly.

size

The size of the queues must match the number of attachment points. Note the special name for the attachment points, for seated passengers the name ends in "sitNN" where NN is any two characters, usually digits.

processes

The input and output settings of the passenger asset. You can specify the amount of input and output for each queue referenced product as well as the duration (or rate) in seconds for that process to take place. This asset spawns or deletes passengers from the model.

passenger_spawn_X

The queue name for the passenger spawn process. The name must be of this form, where "X" is the platform number, starting at 0. Passenger off queues must be named similarly.

string_table

Defines the name for each platform track.

icon0

Window preview icon - see information box [Page 75](#).

Passenger Station config.txt

```
kuid <KUID2:###:####:1>
username Small_Station Example
kind industry
light 1
region TRS2004
type Trackside
icon-texture icon_small_station.tga
script SmallStation
class SmallStation
passenger-height 1.204
kuid-table {
  passenger <kuid:-3:10060>
}
mesh-table {
  default {
    mesh small_station.im
    auto-create 1
    effects {
      0 {
        kind name
        fontsize 0.08
        fontcolor 220,220,220
        att a.name0
        name name
      }
      1 {
        kind name
        fontsize 0.08
        fontcolor 220,220,220
        att a.name1
        name name
      }
      2 {
        kind name
        fontsize 0.08
        fontcolor 220,220,220
        att a.name2
        name name
      }
    }
  }
}
attached-track {
  track_one {
    track <kuid:-1:15>
    vertices {
      0 a.track0a
      1 a.track0b
      2 a.track0c
      3 a.track0d
    }
  }
  track_two {
    track <kuid:-1:15>
    vertices {
      0 a.track1a
      1 a.track1b
      2 a.track1c
      3 a.track1d
    }
  }
}
attached-trigger {
  trigger_track_one_a {
    att a.track0a
    radius 2
    track track_one
  }
  trigger_track_one_b {
    att a.trigmain
    radius 19
    track track_one
  }
  trigger_track_one_c {
```

```

        att    a.track0d
        radius  2
        track  track_one
    }
    trigger_track_two_a {
        att    a.track1a
        radius  2
        track  track_two
    }
    trigger_track_two_b {
        att    a.trigmain
        radius  19
        track  track_two
    }
    trigger_track_two_c {
        att    a.track1d
        radius  2
        track  track_two
    }
}
queues {
    passengers_on_0 {
        passenger-queue 1
        size 48
        initial-count 0
        product-kuid <kuid:-3:10060>
        attachment-points {
            0 a.passon30
            1 a.passsit02
            2 a.passsit03
            3 a.passsit04
            4 a.passon32
            5 etc
        }
    }
    passengers_on_1 {
        passenger-queue 1
        size 40
        initial-count 0
        product-kuid <kuid:-3:10060>
        attachment-points {
            0 a.passon61
            1 a.passsit67
            2 a.passon43
            3 etc
        }
    }
    passengers_off_0 {
        passenger-queue 1
        size 26
        initial-count 0
        product-kuid <kuid:-3:10060>
        attachment-points {
            0 a.passoff01
            1 a.passoff15
            2 a.passoff03
            3 etc
        }
    }
    passengers_off_1 {
        passenger-queue 1
        size 26
        initial-count 0
        product-kuid <kuid:-3:10060>
        attachment-points {
            0 a.passoff27
            1 a.passoff44
            2 a.passoff29
            3 etc
        }
    }
}
processes {
    passenger_spawn_0 {
        start-enabled 1
        duration 20
        outputs {
            0 {

```

```

                amount 1
                queue passengers_on_0
            }
        }
    }
    passenger_spawn_1 {
        start-enabled 1
        duration 20
        outputs {
            0 {
                amount 1
                queue passengers_on_1
            }
        }
    }
    passenger_delete_0 {
        start-enabled 1
        duration 3
        inputs {
            0 {
                amount 1
                queue passengers_off_0
            }
        }
    }
    passenger_delete_1 {
        start-enabled 1
        duration 3
        inputs {
            0 {
                amount 1
                queue passengers_off_1
            }
        }
    }
}
soundscript {
    dayloop {
        repeat-delay 0
        distance 8,130
        sound {
            station_amb_2.wav
        }
    }
}
string-table {
    smallstation_plat1 Platform 1
    smallstation_plat2 Platform 2
}
username-fr Gare_petite
string-table-fr {
    smallstation_plat1 Plateforme 1
    smallstation_plat2 Plateforme 2
}
username-it Small_Station
string-table-it {
    smallstation_plat1 Platform 1
    smallstation_plat2 Platform 2
}
username-de Kleiner_Bahnhof
string-table-de {
    smallstation_plat1 Bahnsteig 1
    smallstation_plat2 Bahnsteig 2
}
username-es Estación_Pequeña
string-table-es {
    smallstation_plat1 Platform 1
    smallstation_plat2 Platform 2
}
}
icon0 <kuid:-3:10164>

```


PASSENGER VEHICLE ASSET

A passenger vehicle asset allows loading and unloading of passengers. No additional script is required for this asset to function. As this is configured to operate with the Passenger Station Asset, an example config.txt file is shown here.

For more information refer to the Passenger Station and Vehicle Assets Tutorial available for download:

http://files.auran.com/TRS2004/downloads/contentcreation/SP2-Passenger_Asset_Tutorial.zip

Breakdown of Passenger Vehicle Config.txt:

Some config.txt tags are not explained below. Others are covered in the general config.txt explanation, see [Page 12](#).

soundscript

Optional sound script for door operation, triggered as an event from the animation file.

mesh

The model name. The model uses an LOD mesh (modelname.lm).

shadow

The model 'shadow' name, the mesh has been included in a subdirectory.

left-passenger-door, right-passenger-door

The 'animated door' names, these must be named like this to work with passenger stations. The mesh has been included in a subdirectory.

.kin

The animation file for the door model, the mesh has been included in a subdirectory.

att-parent 'name'

The door mesh attachment point is located within the mesh 'name'.

queues

The queues field defines the passenger product, the size and the initial count when placed. Passenger attachment points placed in 3dsmax/gmax are referenced, but only a limited number have been included in this example.

passengers

The name you choose for the queue.

size

The size of the queues must match the number of attachment points, note the special names, for seated passengers, "a.sitNN", for standing passengers, "a.standNN", where NN is any two characters, usually digits. Each point must have a unique name.

initial-count

The initial number at starting.

passenger-queue

Enables the queue to carry passengers, when set to 1.

product-kuid

The passenger product kuid required for this vehicle.

Passenger Vehicle Config.txt

```
kuid <KUID2:###:####:1>
name BR MK1 RMB Example
company British Railways
kind traincar
region TRS2004
kuid-table {
}
engine 0
mass 28000
bogey <kuid:-3:10061>
enginespec <kuid:-1:42004201>
soundscript {
    door_open {
        trigger door_open
        nostartdelay 1
        repeat-delay 1
        distance 5,170
        sound {
            start.wav
        }
    }
    door_close {
        trigger door_close
        nostartdelay 1
        repeat-delay 1
        distance 5,170
        sound {
            start.wav
        }
    }
}
mesh-table {
    default {
        mesh mk1_rmb_body/mk1_rmb_body.lm
        auto-create 1
    }
    shadow {
        mesh mk1_rmb_shadow/mk1_rmb_shadow.pm
    }
    left-passenger-door {
        mesh mk1_rmb_body/left_door/left_door.im
        anim mk1_rmb_body/left_door/left_door.kin
        auto-create 1
        att a.doors
        att-parent default
    }
    right-passenger-door {
        mesh mk1_rmb_body/right_door/right_door.im
        anim mk1_rmb_body/right_door/right_door.kin
        auto-create 1
        att a.doors
        att-parent default
    }
}
queues {
    passengers {
        size 44
        initial-count 0
        passenger-queue 1
        product-kuid <kuid:-3:10060>
        attachment-points {
            0 a.sit5a
            1 a.sit1b
            2 a.stand3c
            3 etc
        }
    }
}
icon0 <kuid:-3:10164>
description "BR MK1 RMB Example Asset"
name-fr "MK1 RMB de BR"
name-de "BR MK1 RMB"
name-es "BR - MK1 RMB"
name-it "BR MK1 RMB"
```


KIND: FIXEDTRACK

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Fixedtrack\

A fixedtrack in TRS could be likened to a model train sectional track system. They snap into position when moved onto another track in Surveyor.

Technically, all a fixedtrack comprises is a mesh asset with an attached track (or tracks) and surveyor only rendered arrows so the user knows where the fixedtrack starts and ends.



The model simply comprises of a few attachment points (using the *a.name* naming convention) set-up accurately in 3dsmax or gmax, and a single invisible polygon to allow exporting, and for in-game asset selection.

Note that correct track end attachment orientation is essential. The Y axis must point 'out' at the correct angle. The Z axis must point 'up'. Mid points only need to be in the correct spline path. See diagram below.

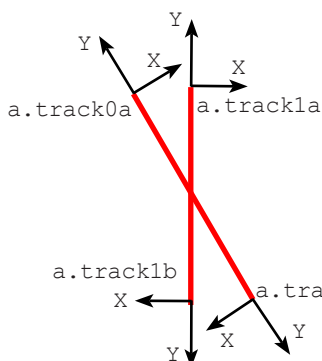
TRS released fixedtracks comprise of only curved and straight sections. Junctions are not possible in TRS, as the lever switching functionality for trains is not implemented. Crossings may be made, just create two attached-track fields.

When a spline track is attached to a fixedtrack the fixedtrack will update to the attached track type. (unless *useadjoiningtracktype 0* is used as shown left)

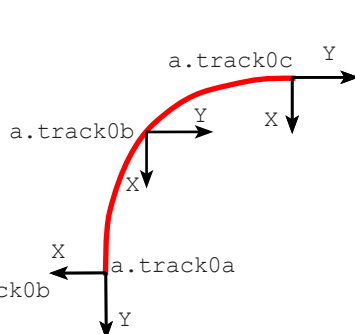
The arrows are inserted at each end as a *kind attachment* - referenced by the arrow's KUID: <KUID:-3:10092>

Each fixedtrack asset needs a preview-mesh, as spline tracks will not render in the Preview window. A preview-mesh can simply be setup as a *kind mesh*. This way the preview-mesh will never be selectable or seen in Surveyor.

Crossing Attachments



Curve Attachments



```
username FT 10 Deg 700m Rad
kind fixedtrack
kuid <KUID2:#####:#####:1>
region Fixed track
type Fixed track
preview-mesh-kuid <KUID:-3:10099>
mesh-table
{
  default
  {
    mesh 10-700.im
    auto-create 1
    effects
    {
      arrow0
      {
        kind attachment
        att a.track0a
        default-mesh <KUID:-3:10092>
        surveyor-only 1
      }
      arrow1
      {
        kind attachment
        att a.track0e
        default-mesh <KUID:-3:10092>
        surveyor-only 1
      }
    }
  }
}
attached-track
{
  track0
  {
    track <KUID:-1:15>
    vertices
    {
      0 a.track0a
      1 a.track0b
      2 a.track0c
    }
  }
}
```

Default fixedtrack preview mesh KUIDs:
STRAIGHT <KUID:-3:10154>
CURVE <KUID:-3:10099>

Crossing Fixed Track

```
attached-track
{
  track0
  {
    track <KUID:-1:15>
    useadjoiningtracktype 0
    vertices
    {
      0 a.track0a
      1 a.track0b
    }
  }
  track1
  {
    track <KUID:-1:15>
    useadjoiningtracktype 0
    vertices
    {
      0 a.track1a
      1 a.track1b
    }
  }
}
```

KIND: TRACK – RAILS

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\track\

This is used for creating rails.

Config.txt :

```
kuid <KUID2:####:#####:1>
kind track
region Britain
type Rails
rgb 255,200,0
length 4
istrack 1
width 4
chunky_mesh mstand_tex
chunky_info 0, 2, 1.2, 0.2, 0.85, 0.3, 0.7
username
description " "
trainz-build 1.5
category-class TR
category-region-0 UK
category-era-0
```

Breakdown:

region

Surveyor region.

type

Surveyor type.

rgb

Color used for display in mini map.

length

Length of track piece.

istrack

Sets whether the track is a rail for trains or not.

1 = This is a rail track

width

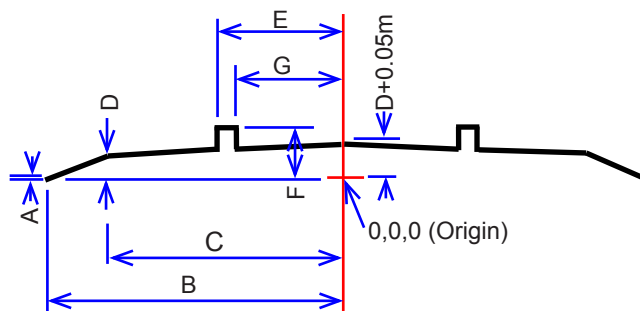
Width of track in meters.

chunky_mesh

Name of texture to apply to rail.

chunky_info

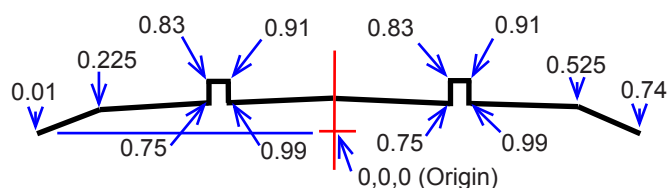
These values (in metres) define the shape of the mesh created for the track. See drawing below:



chunky_info A, B, C, D, E, F, G
chunky_info 0, 2, 1.2, 0.2, 0.85, 0.3, 0.7

chunky_info texture file

The texture file with the track texture on the left and a rail texture on the right is 128 x 128 uncompressed tga, and may have an alpha layer. The texture is mapped to the mesh shape above using the values in the drawing below, as fractions of the 128 pixel width.



Fraction	0.01	0.225	0.525	0.74	0.75	0.83	0.91	0.99
Pixels	1	29	67	95	96	106	117	127

KIND: TRACK – ROAD

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Track\

This is used for creating roads.

config.txt:

```
kuid <KUID2:####:####:1>
kind track
type Roads
region Australia
length 5
grounded 0.4
istrack 0
width 7.9
bendy 1
isroad 1
uncached_alphas 1
carrate 55
obsolete-table
{
}
username
description " "
trainz-build 1.5
category-class SR
category-region-0 AU
category-era-0
```

Breakdown:

region

Surveyor region.

type

Surveyor type.

length

Length of track segment in meters.

grounded

Height in meters for the road to be offset from terrain.

istrack

0 = This is not rail tracks .

width

Width of track mesh in meters.

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bent around corners.

isroad

Specifies track is a road with cars,

1 for cars to appear on road

0 for no cars to appear.

uncached_alphas

This is used in certain situations to improve alpha sorting. This should only be set to 1 for tracks that use an alpha texture and are always placed flat near the ground. This is only used on road splines.

carrate

Defines traffic density on road (minimum seconds between each car generated).

0 = No traffic.

Number must be greater than 3.

Refer to *Tips and Tricks Chapter* on how to add created vehicles to road traffic.

Additional Notes for splines:

Splines may be used to create a number of objects that are not track, a building, bridges, walls, fences a row of poles or trees for example.

upright 0

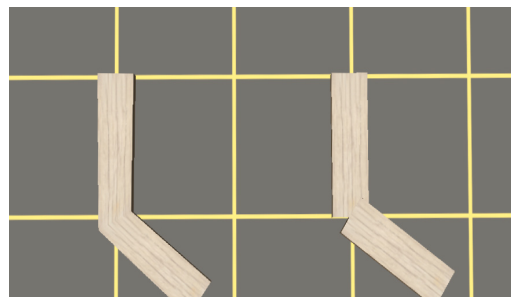
This effects how vertical the objects in the spline are, for example a row of poles:

0 = the poles will be placed at right angles to the slope of the ground.

1 = the poles will be truly vertical regardless of the ground slope.

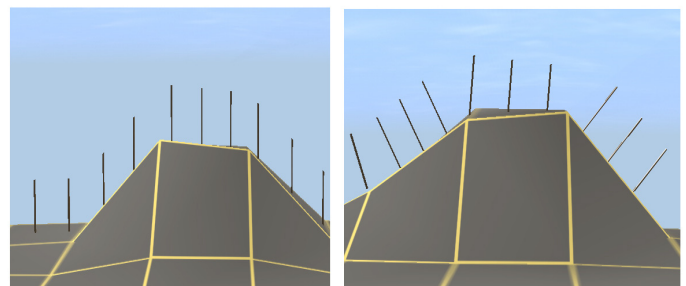
Notes: bendy and upright have a visible effect for Kind Track splines, see diagram below. For Kind Bridge or Tunnel, the splines show as in bendy 1, bendy 0 has no visual effect. However, bendy 1 should always be entered in the config.txt file for bridge and tunnel Kinds, as the tag improves handling of the spline and Trainz performance .

A Kind track placed on the ground is now height adjustable, and is the best option for most spline models. Splines of Kind track have a much less frame rate and performance impact than Kind bridge.



Bendy 1

Bendy 0



Upright 1

Upright 0

KIND: BRIDGE – SINGLE TRACK

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\track\

or

C:\Program Files\Auran\TRS2004\World\Custom\splines\

This kind is used for creating road and rail bridges.

Config.txt

```
kuid <KUID2:####:#####:1>
kind bridge
type Bridges
region Britain
length 20
bendy 1
bridgetrack <KUID:-1:100395>
trackoffsets 0.01
height -8
casts_shadows 1
istrack 1
initiator dark_stone_arch_2t_start
terminator dark_stone_arch_2t_end
endlength 40
kuid-table {
    0 <KUID:-1:100395>
}
username
description " "
trainz-build 2.4
category-class TB
category-region-0 UK
category-era-0
```

Breakdown:

type

Surveyor type – e.g. bridge, tunnel or rail.

region

Surveyor region.

length

Length in meters of each bridge piece.

bridgetrack

Kuid for the type of rail or road used on bridge.

trackoffsets

Distance in meters the rail/s are attached to the center of the bridge spline. Any number of tracks can be attached to the spline, only splines with the same track offsets can be connected together. For a single track bridge a small offset of 0.01 metres for example, is necessary.

height

Height from the track level to the base of the bridge supports, should be negative for bridges.

casts_shadows

Defines whether or not the shadows are cast.

0 = shadows off

1 = shadows on

If shadows are on there needs to be a bridge_shadow.im model in a sub folder, for the bridge spline model, and the initiator and terminator segments (if they are used).

istrack

0 = This is a road bridge

1 = This is a rail bridge

Initiator

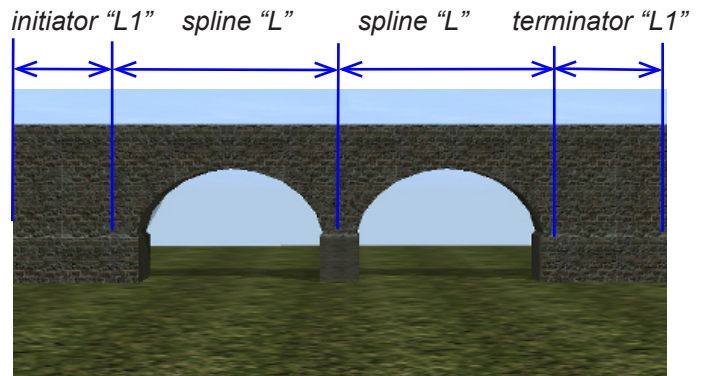
Name of model to use at start of bridge, placed in sub folder with same name.

terminator

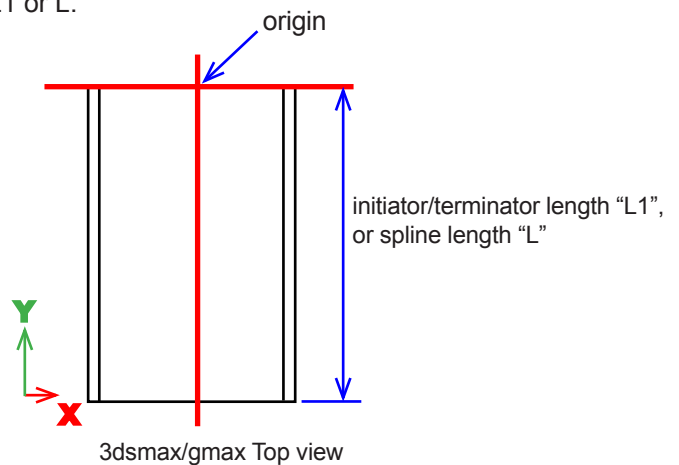
Name of model to use at end of bridge, placed in sub folder with same name.

endlength

Length in meters of the initiator and terminator models.

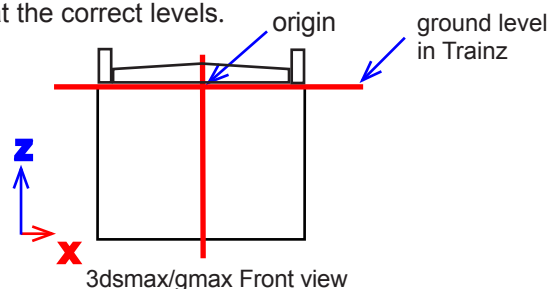


In 3dsmax/gmax, the initiator, terminator and spline models must be constructed starting on the origin and extending in the negative Y axis direction. The top view in the diagram shows the correct placement and dimensions L1 or L.



The initiator may be rotated 180 degrees to create a terminator model, if required.

Attachment points will be automatically generated in Trainz at ground level. The model heights need to be adjusted in 3dsmax/gmax so a road or track will connect at the correct levels.



For additional notes on splines refer to [Page 84](#).

KIND: BRIDGE – TUNNEL

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\track\

or

C:\Program Files\Auran\TRS2004\World\Custom\splines\

This kind is used for creating road and rail tunnels.

Config.txt :

```
kuid <KUID2:####:#####:1>
kind bridge
type Tunnels
region Australia
length 20
bendy 1
bridgetrack <KUID:####:#####>
trackoffsets -2.5, 2.5
height 8
istrack 1
initiator oz_tunnel_start
terminator oz_tunnel_end
endlength 20
kuid-table
{
  0 <KUID:####:#####>
}

username
description " "
trainz-build 2.4
category-class TB
category-region-0 AU
category-era-0 2000s
```

istrack

0 = This is a road bridge (tunnel)

1 = This is a rail bridge (tunnel)

Initiator

Name of model to use at start of bridge, placed in subfolder with same name.

terminator

Name of model to use at end of bridge, placed in subfolder with same name.

endlength

Length in meters of the initiator and terminator models. Refer to [Page 85](#) for information on constructing initiator and terminator models.

For additional notes on splines refer to [Page 84](#).

Breakdown:

type

Surveyor type – e.g. bridge, tunnel or rail.

region

Surveyor region.

length

Length in meters of each bridge (tunnel) segment.

bridgetrack

Kuid for the type of rail or road used on the bridge.

trackoffsets

Distance in meters the rail/s are attached to the center of the bridge spline. Any number of tracks can be attached to the spline, only splines with the same trackoffsets can be connected together.

For a single track tunnel, use:

trackoffsets 0.01

height

The height value for tunnels should be positive and greater than the height of the ceiling of the tunnel, but less than the height of the tunnel entrance structure.

KIND: BRIDGE – DOUBLE TRACK

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\track\

This kind can also be configured to create splines that can be used for placing two or more tracks using the *trackoffsets* tag.

Config.txt :

```
kuid <KUID2:####:#####:1>
kind bridge
type Rails
region Australia
length 20
bendy 1
bridgetrack <KUID:-1:100396>
trackoffsets -2.5,2.5
height 0
istrack 1
kuid-table {
    0 <KUID:-1:100396>
}
username
description " "
trainz-build 2.4
category-class TB
category-region-0 AU
category-era-0 2000s
```

Breakdown:

type

Surveyor type – e.g. bridge, tunnel or rail.

region

Surveyor region.

length

Length in meters of each bridge segment.

bridgetrack

Kuid for the type of rail used on bridge.

trackoffsets

Distance in meters the rail/s are attached to the center of the bridge spline. Any number of tracks can be attached to the spline, only splines with the same track offsets can be connected together.

height

0 is used for double tracks.

istrack

0 = This is a road bridge

1 = This is a rail bridge

Refer to [Page 85](#) for information on constructing initiator and terminator models, if required.

For additional notes on splines refer to [Page 84](#).

KIND: MOSPEEDBOARD

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\trackside\

This is a speed limit sign.

Config.txt:

```
kuid <KUID2:####:#####:1>
kind mospeedboard
region Australia
trackside -2.5
speedlimit 5.56
username
description " "
trainz-build 2.4
category-class WS
category-region-0 AU
category-era-0 2000s
```

Breakdown:

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

Speedlimit

This value is the maximum speed allowed in meters per seconds.

Note:

To convert miles per hour to meters per second multiply by a conversion factor of 0.447.
e.g. 10mph is 4.47 m/s.

To convert kilometers per hour to meters per second multiply by a conversion factor of 0.278.
e.g. 10kph is 2.78m/s.

KIND: MOSIGNAL

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\trackside\

The standard light signals in TRS.

Config.txt

```
kuid <KUID2:####:#####:1>
kind mosignal
mesh-table
{
  default
  {
    mesh signalname.im
    auto-create 1
    effects
    {
      0
      {
        kind name
        fontsize 0.15
        fontcolor 30,30,30
        att a.name0
        name name
      }
    }
  }
}
light 1
trackside -2.7
function TrackSignal
region Britain
obsolete-table
{
}
username
description " "
trainz-build 2.4
category-class WA
category-region-0 UK
category-era-0
```

Breakdown:

region

Surveyor region.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is light by general light value (uniform colouring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

function

Must be set to TrackSignal

Note: All editable signage must use the effect - name set-up when a mesh-table is used. Refer Effects on [Page 13](#).

SIGNALS

The next section of the config explains which aspects the signal is capable of displaying, and also which light points are activated when each state is displayed. The supported aspects are configured by reference number as follows...

0	STOP
1	STOP THEN PROCEED
2	CAUTION AND LEFT DIVERGE
3	CAUTION AND RIGHT DIVERGE
4	CAUTION
5	PROCEED AND LEFT DIVERGE
6	PROCEED AND RIGHT DIVERGE
7	ADVANCED CAUTION
8	PROCEED

The following two aspects are only used for scenarios....

9	SLOW
10	MEDIUM SPEED

The aspect section of the config.txt is arranged as follows.....

```
signals
{
  0
  {
    light 7
  }
  2
  {
    light 6,0,1,2,3,4
  }
  4
  {
    light 6
  }
}
```

(don't forget the 'space' between the number and bracket)

Looking at the example above, under the heading 'signals' we see the states the signal is capable of displaying in the left column. From this extract we can see that this signal is capable of displaying aspects 0, 2 & 4.

When aspect 0 (stop) is displayed, light point 7 is activated.

When aspect 2 (caution left) is displayed, light points 6,0,1,2,3,4 are activated

When aspect 4 (caution) is displayed, light point 6 is activated.

LIGHTS

Each light point needs to have a corona associated with it. Coronas are stored in each signal object's directory alongside it's textures. Examples have been packaged within the zip file this document was located.

```
lights
{
  0
  {
    corona corona_white.tga
  }
  1
  {
    corona corona_white.tga
  }
  2
  {
    corona corona_white.tga
  }
  3
  {
    corona corona_white.tga
  }
  4
  {
    corona corona_white.tga
  }
  5
  {
    corona corona_green.tga
  }
  6
  {
    corona corona_yellow.tga
  }
  7
  {
    corona corona_red.tga
  }
  8
  {
    corona corona_white.tga
  }
  9
  {
    corona corona_white.tga
  }
}
```

(don't forget the 'space' between the number and bracket)

Looking at the example above, under the heading 'lights' we see the light points that are attached to the 3D model. This model has 10 of them, they are named a.light0 to a.light9.

From the signals section we know that when aspect 0 (stop) is displayed, light point 7 is activated.

Looking at the extract left...

When light point 7 is activated, it displays corona red.

When aspect 2 (caution left) is displayed, light points 6,0,1,2,3,4 are activated

When light point 6 is activated, it displays corona_yellow.

When light points 0 – 4 are activated, each displays corona_white.

Signal placement is very important for correct operation of the system. There are some rules to consider while signaling your map which if not observed may cause problems with getting the correct aspects to display.

There are also various departures from prototypical operation which should be considered when designing new signaling, and also when installing it into a map.

Please refer to the Signaling document for further details.

Default location:

C:\Program Files\Auran\TRS2004\Docs>manuals_cd\
manuals\Signaling_Guide.pdf

KIND: MOJUNCTION

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\trackside\

This is used for creating junction control levers.

Config.txt :

```
kuid <KUID2:####:#####:1>
kind mojunction
mesh-table
{
  mode0
  {
    mesh lever1/lever1.im
    auto-create 1
  }
  mode1
  {
    mesh lever2/lever2.im
    auto-create 1
  }
}
region Australia
trackside 2
light 1
soundscript
{
  toggle
  {
    trigger toggle
    distance 5, 100
    nostartdelay 1
    repeat-delay 1
    sound
    {
      points.wav
    }
  }
}
obsolete-table
{
}
username
description " "
trainz-build 2.4
category-class WX
category-region-0 AU
category-era-0
```

Breakdown:

region

Surveyor region.

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will place the object on the right.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is light by general light value (uniform colouring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

mode0

The model name of the initial junction, the model file is located in subfolder (lever1). Example refers to a file lever1\lever1.im

mode1

The model name of the initial junction, the model file is located in subfolder (lever2). Example refers to a file lever2\lever2.im

Note that this model does not have an animation file. The model changes between the two positions defined by the lever1.im and the lever2.im file models.

soundscript

Soundscripts for mojunction objects can be activated with toggle trigger message as in the example.

When a model has animation it is possible to trigger effects by an event associated with the animation at defined key frames of the animation. This event is activated by a trigger name such as "toggle" in the .evt file and called up from the config.txt file.

Refer to the Animation Event descriptions on [Page 104](#).

However, for this model without an animation, the trigger name "toggle" is recognised by Trainz when the mojunction is activated, and the sound will play.

If you delete the Auran default junction lever for a switch junction in Trainz and replace it with your model, the wave file defined in the config.txt file will be associated with the trigger and play instead of the default in-built sound.

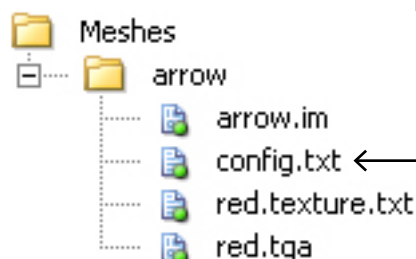
KIND: MESH

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\meshes\

These are meshes that are never referenced through Surveyor panels but are referenced by KUID from another asset.

It could be referenced through the *preview-mesh-kuid* tag (like the airport does) or as a *kind attachment* effect like that of fixed-tracks.



Typical mesh config.txt

```
kuid <KUID:-3:10092>
kind mesh
mesh-table
{
  default
  {
    mesh arrow.im
    auto-create 1
  }
}
```

KIND: MOTURNTABLE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenery\

This is a turntable object.

Config.txt :

```
kuid <KUID2:####:#####:1>
kind turntable
region Australia
type Trackside
mesh-table
{
  default
  {
    mesh turntablebase/turntablebase.im
    auto-create 1
  }
  turntable
  {
    mesh platform/platform.im
    auto-create 1
  }
}
light 1
mode0 oz_turntable2
model oz_turntable2_spinner
angle 0,165,180,345
track <KUID:-1:100966>
snapmode 2
dighole 3,3
kuid-table
{
  0 <KUID:####:#####>
}
obsolete-table
{
}
username
description " "
trainz-build 2.4
category-class TR
category-region-0 AU
category-era-0
```

Breakdown:

type

Surveyor type.

region

Surveyor region.

default (Previously mode0 for pre-TRS releases)

The name of the main turntable object, now referenced from the mesh-table.

turntable (Previously mode1 for pre-TRS releases)

Name of the rotating turntable part, now referenced from the mesh-table.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is lit by general light value (uniform colouring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

angle

Specifies the angles at which the turntable stops.

NOT USED IF THE TURNTABLE IS SET UP AS AN ANIMATION SEE BELOW.

track

Kuid for track to be attached to turntable

snapmode

Specifies the alignment of the turntable to the surveyor grid. 1 = origin snaps to grid (use for removing even dighole values), 2 = origin snaps to the center of a grid square (use for odd dighole values).

dighole

Specifies the number of grid segments (length, width) to be removed from the surveyor grid to accommodate the turntable pit.

ANIMATED TURNTABLES

Turntables can now be set up as an animation.

Keyframes can be specified as the stopping points much like 'angles' above. Use attached-tracks at keyframe points.

keyframes

Specifies where on the animation the turntable is to stop.

frame-rate

Generally make this 30

Example:

The example below shows the entries using an animation file for a transfer table model. For a sample model, see:

<http://www.auran.com/TRS2004/downloads/contentcreation/TransporterTestAsset.zip>

```
kuid <KUID:44179:60004>
region Australia
light 1
kind turntable
type Trackside
username transporter
track <KUID:-1:100966>
snapmode 1
dighole 4,8
keyframes 0,100,201,300,400,601
looping 0
frame-rate 30
nightmode home
interior <KUID:44179:55002>
mesh-table
{
  default
  {
    mesh trans_base/trans_base.im
    auto-create 1
  }
  turntable
  {
    mesh trans_platform/trans_platform.im
    anim trans_platform/anim.kin
  }
}
Etc.
```

KIND: MOCROSSING

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenery\

This is an example level crossing consisting of script animated boom gates, and flashing lights, however, only the config.txt entries are discussed here. Some duplicate entries in the config.txt file for the corona effects are not shown to conserve space.

Note that an animated mocrossing can be created without the use of scripts, the animation being triggered automatically by a train. Animation events may also be linked to an animation key-frame to give control over effect and script timing.

The track and road types are specified in the attached-track and the meshes are within the mesh-table. An invisible road KUID may be useful for this type of model.

Breakdown:

class

Script class.

script

Script file name.

kuid-table

Lists the dependencies (track, road, corona)

mesh

Name of the default mesh model.

anim

Animation file attached to the default mesh.

att

Attachment point for the sub mesh.

att-parent

Defines the mesh to which the sub mesh is attached.

auto-create 1

The mesh is visible when the model is placed.

effects

The attached effects (coronas).

directional 0

The default alignment is overridden so the corona always faces the screen.

texture-kuid

KUID for the corona.

attached-track

Track type that will be joined between the attachment points, The names "a.track" and "a.road" are not mandatory, any name using the "a.name" convention may be used. The attachment points are defined in the model mesh. Refer to [Page 82](#) for attachment orientation.

useadjoiningtracktype 0

prevents the track used being updated to match the attached track.



config.txt

```
kuid    <KUID2:#####:#####:1>
light   1
type     trackside
kind     mocrossing
trainz-build 2.4
category-region-0    AU
category-era-0       1980s
region              Australia
class               modular_xing
script              xing.gs
KUID-table {
    corona_red      <KUID:-3:10112>
    road            <KUID:###:#####>
}
mesh-table {
    default {
        mesh        road.im
        anim boomgates.kin
        auto-create  1
        effects {
            pole1-light1 {
                kind corona
                att a.pole1-lamp0
                texture-kuid <KUID:-3:10112>
            }
            pole1-light2 {
                kind corona
                att a.pole1-lamp1
                texture-kuid <KUID:-3:10112>
            }
            boom1-light1 {
                kind corona
                att a.boom1-light0
                directional 0
                texture-kuid <KUID:-3:10112>
            }
            boom1-light2 {
                kind corona
                att a.boom1-light1
                directional 0
                texture-kuid <KUID:-3:10112>
            }
        }
    }
}
attached-track {
    track {
        track          <KUID:-1:15>
        vertices {
            0          a.track0a
            1          a.track0b
        }
    }
    road {
        track          <KUID:###:#####>
        useadjoiningtracktype 0
        vertices {
            0          a.road0a
            1          a.road0b
        }
    }
}
string-table {
}
username      QR Level Crossing
category-class TR
description    "QR level crossing with animated boomgates and flashing lights."
```

string-table

A list that can be accessed in script/scriptlet code.

KIND: ACTIVITY

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenarios\

An activity is a scripted scenario.

Config.txt :

```
kind activity
kuid <KUID:-14:160>
username Highland Valley (DCC)

scriptlibrary SP3S1DCC
scriptclass SP3S1DCC

driver-settings
{
  autopilotmode 0
  startingtime 0.4
  timerate 1
  deraillevel 0
  showhelp 0
  controlmethod 0
  weather 3
  changeability 1
}
kuid-table
{
  highland_valley <KUID:-12:132>

  f7_sfired <KUID:-1:1>
  atsf_chair <KUID:-1:100160>
  atsf_pullman_pine <KUID:-1:100163>
  atsf_baggage <KUID:-1:100159>
  cflow_fert <KUID:-1:100012>
  prr_fm_tuscan <KUID:-1:100017>
  40ft_boxcar <KUID:-1:100085>
  pdhc_babyruth <KUID:-1:100066>
  4bhopper_il <KUID:-1:100929>
  50ft_boxcar <KUID:-1:100086>
  gatx_pennsalt <KUID:-1:100092>
  60ft_boxcar <KUID:-1:100087>
  sd40_2_santafe <KUID:-1:100871>
  4bhopper_il_coal_full <KUID:-1:101224>
  foundry_car <KUID:-1:101220>
}
description "Take contol of the morning
passenger service to Highland Valley stopping
at all stations and return to Greenwood. Bad
weather is forecast so drive with care.

Service : Highland Valley Passenger
Train No. : 7528
Consist : F7A + 5 cars
Weight in Tow : 300t
Total Length : 490'"
```

Breakdown:

username

Name of scenario displayed in TRS.

scriptlibrary

The name of the .gsl (compiled script) library on disk, without the ".gsl" extension.

scriptclass

The name of the scenario class within the script file.

driver-settings{}

Specify the settings of this scenario, similar to Driver's 'settings' screen. Allows you to set the weather, control method (0 – dcc, 1 – cabin controlled) etc.

<i>autopilotmode</i>	<i>0=off 1=on</i>
<i>startingtime</i>	<i>0..1 (0.5=midday)</i>
<i>timerate</i>	<i>1=real-time</i>
<i>deraillevel</i>	<i>0=none 1=arcade 2=realistic</i>
<i>showhelp</i>	<i>0=off 1=on</i>
<i>controlmethod</i>	<i>0=dcc 1=cabin</i>
<i>weather</i>	<i>0=clear 1=cloudy 2=drizzle 3=rain 4=stormy 5=light snow 6=medium snow 7=heavy snow</i>
<i>changeability</i>	<i>0=none 1=periodic 2=extreme</i>

kuid-table{}

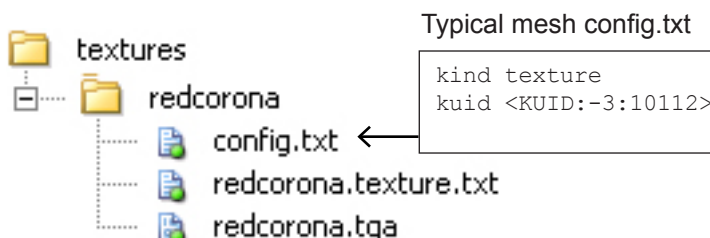
A list of named assets used in the scenario. Scripts refer to assets (eg trains) by the names in this table.

KIND: TEXTURE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\textures\

You can now reference a simple texture as an asset. These can be referenced by kuid for use as a custom corona for example.



KIND: BUILDABLE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\scenery\

A Kind buildable is a variant of Kind Scenery, and may be used in place of a normal scenery item, with the following attributes:

It inherits the Kind scenery attributes:

It allows attached track to be used as part of the model, refer to [Page 82](#) for information of the attachments and orientation;

It does not support processes, use Kind Industry for this purpose.

MISCELLANEOUS CONFIG.TXT TAGS

There are a number of miscellaneous tags used in config.txt files that may not have been covered fully in previous Kind examples. A brief summary is provided for some of these here:

invisible 1

A tag only used for track, the track will only show in Surveyor.

*For other meshes use **surveyor-only 1** to make it invisible in Driver, refer to [Page 82](#) for an example.*

trigger 1

Used for a trackside object to send script messages.

default-night-forward, default-night-reverse

This is the name for a sub mesh attached to a locomotive, to show a beam of light for example, in the direction of movement of the locomotive. Trainz recognises the name and turns on the correct mesh depending on the running direction.

Note in the example the use of auto-create 0 to make the mesh invisible when placed. The mesh will be visible when the light switch is activated.

default-night

Similar to the previous tag, but not dependent on running direction.

use-parent-bounds

```
{
  default
  {
    mesh loco_body/loco_body.lm
    auto-create 1
  }
  default-night-forward
  {
    mesh loco_body/night/night.im
    auto-create 0
    att a.bog0
    att-parent default
  }
}
```

Used for an animated object. A bounding box is a fairly generic term used to describe the size of an object for clipping purposes. If the bounding box goes off the screen, the game will stop rendering the object. The idea is that the bounding box completely wraps the object, but this is not always the case. Animation, for example, may cause the object to pass outside its precalculated bounding box. Distance is not relevant; it's simply a question of whether the bounding box is on the screen or not.

The tag relates the sub mesh bounding box to the main mesh bounding box, and makes the animated sub mesh continuously visible. The tag is placed within the sub mesh config.txt entries

buffer-speed

*Sets the upper speed limit for an object to act as a buffer. The value **buffer-speed 5** sets a 5 metres per second speed as the limit for stopping a traincar. Used in a Kind mosignal object.*

tender 1

Indicates the traincar is a locomotive tender. Used in the tender config.txt file to enable the tender to remain coupled with the locomotive in a runaround movement.

CHAPTER 3

Understanding and using Content Creator Plus

The purpose of this chapter is to introduce users to Content Creator Plus, a powerful program used to generate and validate “config.txt” files. Content Creator Plus is intended for experienced content creators and a degree of prior knowledge regarding kinds and containers is assumed.

Getting Started

Main Screen

The main screen is divided into a number of areas:

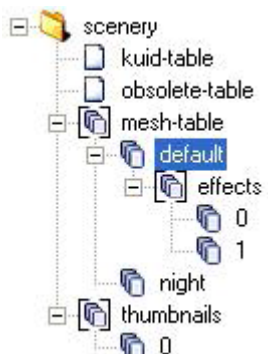


1. Tree View 2. Tag space 3. 3D Viewport 4. Error Message Box.

1. Tree View

When an existing config file is loaded, the program will “parse” the text file and gather 2 different sorts of information: the tags and the containers. The tags are values being assigned to a property, and a container is a section of the config.txt that groups a number of tags or other containers.

With many possible tags and containers, the best way to manage the containers is to load them into a tree view hierarchy (like the tree view of Explorer for example). The diagram below is a section of the tree view for a Kind traincar model:



The top “node” or Kind is called traincar, and is the main container for the complete config file. Traincar is the name of the Kind that we are currently creating (for example, if we were making a bogey, it would be written “bogey”). Under that container are other sub containers in the config file.

The tree may be freely expanded or collapsed (by clicking with the Left mouse button on the plus or minus symbols). When you click on one of the nodes, it will load the tags that are included for that container in the tag space, so you can add new entries or edit existing values. Left clicking on the node icon will select that node, whilst Left clicking the node name will select the node and allow you to rename it.

When you click with the Right mouse button on one of the nodes, a contextual menu will pop up with four different sections:



Sub-Containers: The first section is the available sub-container section. This section shows which sub-containers you may add to the opened container. Select a sub-container by clicking on it with the Left mouse button. Depending on the type of container added, a number of compulsory entry dialog boxes may appear when you now click with the Left mouse button on the new container name.

Tags: The second section is the tag section. This shows you the list of non-compulsory tags (a non-compulsory tag is an optional tag for additional functionality). When you click on one of the choices with the Left mouse button, it will be added to the tag space that represents the container. You may also delete any non-compulsory tags.

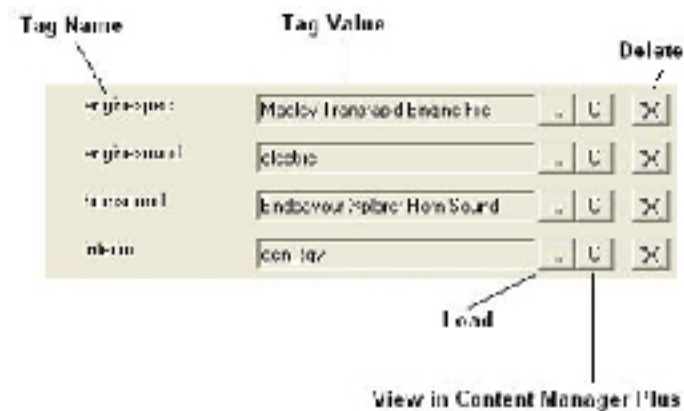
Rename: The rename option allows you to change the name of a container.

Remove: The remove option allows you to delete an unwanted container from the config file. This is particularly useful if you are editing a config.txt from an existing asset to create a new asset. Click on a container name with the Right mouse button to open the options for the container, and select an item using the Left mouse button.

2. Tag Space

This section of the program displays the content of a container. It dynamically changes as you either select different containers or add/delete tags.

Select tags may have a variety of buttons placed adjacently:



Tag Name: The name of every mandatory and/or selected single tag will be displayed on the left hand side of the space.

Tag Value: The value stored in the selected tag.

Load: This button will let you select a file or asset (depending on the tag type).

View in Content Manager Plus: Will open Content Manager Plus and find the selected asset.

Delete: Removes the selected tag from the working config.txt.

3. 3D Viewport

When a mesh is referenced for the model, you need to specify a mesh file. Clicking on the “...” button will allow you to browse for the filename. The program will load the referenced mesh file, and display it in the 3D viewport. This viewport only appears in Content Manager Plus when a valid mesh is loaded.



Buttons below the viewport allow the camera to be moved, rotated or zoomed. Alternatively, you can control the model with your mouse by interacting directly with the 3D Viewport.

Use the Mousewheel to zoom in and out

Hold the Left mouse button down to move

Hold the Right mouse button down to rotate

4. Error Message Box

When you load or save a config.txt file, the program will validate your data input to make sure that everything is correct. The error box is used to output error/warning messages to the user to assist error checking if the model is not working in Trainz.

Clicking on an error message with the Left mouse button will jump to the relevant tag in the program that is causing the error.

You can also Right click on the message, which will bring up a contextual menu that will let you copy the error message to the clipboard.

Using Content Creator Plus

Creating a new Asset

In the “Content Manager Plus Program”, clicking on the  button (New) will open Content Creator Plus and display the “New Asset” window, from which you can select a base Kind to use as a template



This form shows all the asset Kinds supported by Trainz. Click on the kind that you want to create and the program will generate the basic config.txt structure for that kind, using the mandatory tags. You may also add additional optional tags and containers to your config.txt file for a more complex asset.

The Working Directory

When you are creating or editing an asset with Content Creator Plus, all of your content exists in a temporary “working directory” which is located in your “.\editing” folder within your Trainz install. Assets in this folder are considered “open for editing”.

All assets required by your model MUST be placed in your working directory.

You can easily open this folder by selecting “Edit|Edit in Explorer” from the main menu bar.

Formats for entering Tag Data

Tags require different forms of entry depending on the type of data required. Some tags require a simple text string, while others will require more complex data like a Vector, a float list or a Boolean value.

String, floating point numbers or integers

These three data types are simple text or numeric values.

A string is a basic text field, a Floating point number (or float) is a number that includes a decimal point, while and integer is a numeric value with no decimal point (whole number). The value placed in the text box will be directly associated with the tag.

Vector

A vector is a series of values. A vector may have any number of values depending on the type of data it stores, for example a Vector3 will display three separate text boxes; one for each value.

Float list entry

A float list for a tag will take a series of floating point number values.

The list above includes a series of four floats, each of which can be removed if required. Additional values may be added to the list by clicking on the add button.

KUIDs

The KUID entry is made up of three different parts: The user ID, the content ID and the version ID.

KUIDs are handled internally, and are unable to be edited directly.

Boolean Entry

A Boolean entry is a simple true or false value. It is represented by a tick box.

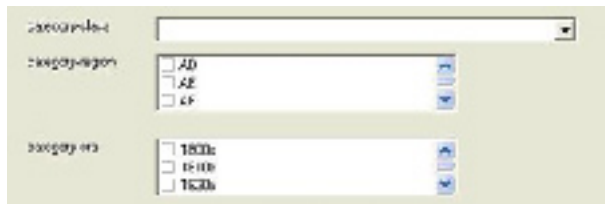
Clicking with the Left Mouse button in the box will toggle

the tick mark on or off. A tick represents “True”.

Data choices

Where a large list of choices is offered, it is convenient to display the available selections in a combo box.

Open the box and select a value by clicking on that value with the Left mouse button. A tick-mark is shown in the box. You may select multiple choices by ticking additional boxes in the list.



File Browser

Clicking on the “...” button adjacent to the appropriate tag with the Left mouse button will open the file browser. The file browser is used to locate the files you wish to use in your asset.

Once the file is located, click on the “Ok” button and the field will be filled with the path for the file you have selected.

Remember that your files must be placed in the working directory.

The file type selections in the browser will reflect the types that are required to fill the dialog box. For example, a mesh dialog box offers a choice between “*.im, *.lm, and *.lm.txt” types, whereas a script will allow files with “*.gs or *.gse” extensions to be selected.

New Assets

Some tags (typically images, scripts or HTML assets) will have a “New” button next to them when they are empty.



Clicking on the “New” button will allow you to create a new asset of the appropriate kind, and will automatically create the necessary files and load them into your editor of choice (specified in your windows settings, for example Photoshop may be your default image editor).

If these tags are populated, the “New” button is replaced with an “Edit” button, which will open the asset in your default editor.

Asset Browser

Some fields require a link to other assets to be created. For simplicity, a list of relevant assets is presented from which the user can select their desired asset.

Browse through the list of installed assets, or type in a partial name in the top search box to locate the asset required. Once you have found the asset you wish to use, Left click on the “Select” button. This will internally store the asset KUID. A null KUID may also alternatively be selected where required.

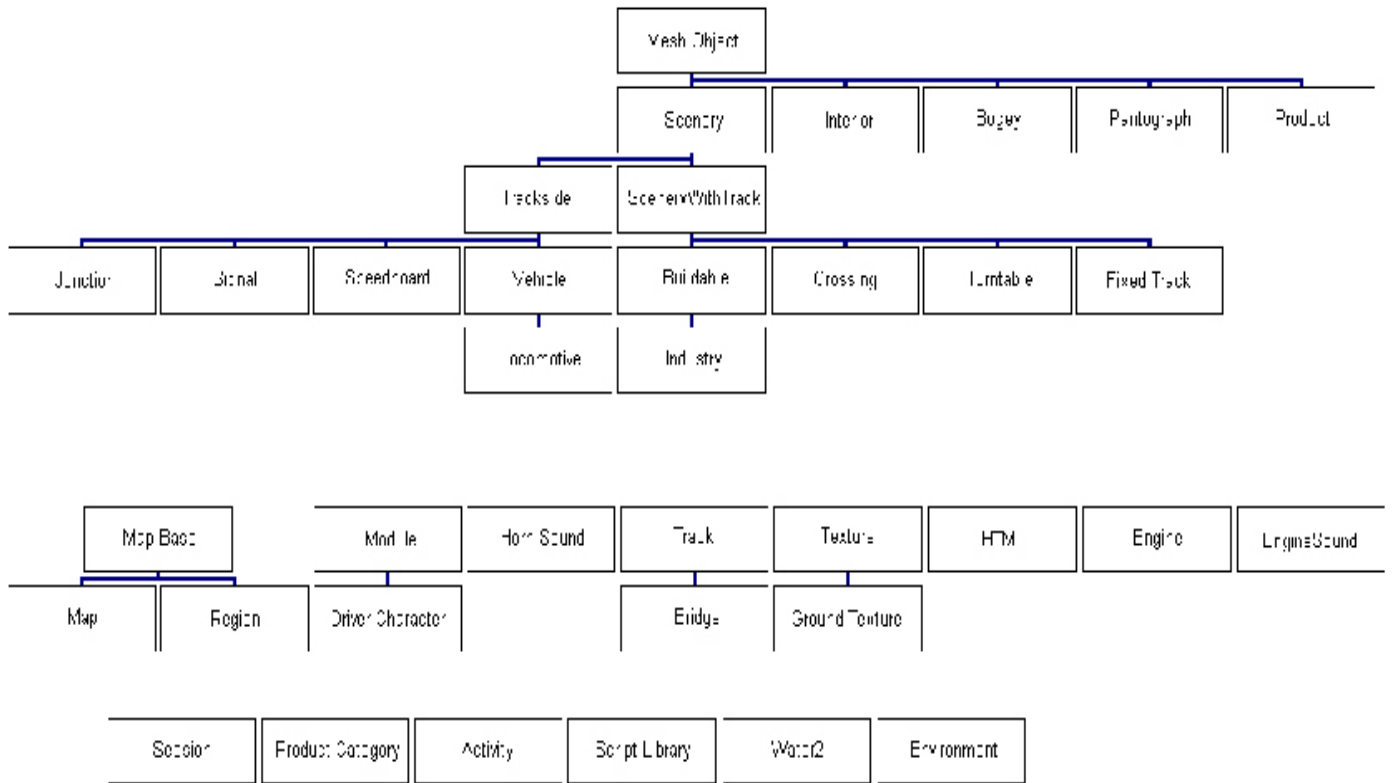


String Asset Browser

The string asset (KUID) browser works similarly to the KUID browser, with the additional ability to change the actual name of the tag.

Inheritance Template

The model asset Kinds have a certain relationship to each other. The following chart shows how these are related, the way the classes are inherited in the game.



CHAPTER 4

Using Content Creator Plus to create a New Asset

The purpose of this chapter is to explain the Workflow process using Content Creator Plus to efficiently create a new asset for Trainz. An example asset will be used to illustrate the process.

Creating a New Asset

The aim of the Content Creator Plus (CCP) module is to create the asset config.txt file. Additionally, it creates a temporary directory where the model files are placed, to be incorporated in the Trainz asset database when the asset is committed.

Please refer to the previous [Chapter 3](#) for an explanation on how to navigate the CCP interface and menus.

A Workflow Process

A logical process should be used to create an asset. This is a brief summary of one such process:

1. Using a pixel editing program, for example Photoshop or Paint Shop Pro, create textures for the new model asset, and save to a temporary directory.
2. Create the mesh model in 3dsmax or gmax, texture and map the model and save the files to the directory.
3. Open CCP and choose the New asset option to select an appropriate Kind for the model, in the example used in this chapter, it would be a Kind
4. Enter as much data as possible, description, username for example, following the guidelines in the previous [Chapter 3](#) for selecting containers and tags, and entering values.
5. Find the newly created directory for the asset, it will be called "New Asset" or in some instances may have a coded name until the config file is first saved and committed with a relevant Username tag. The "New Asset" will also appear in the Content Manager Plus main screen.
6. Make any appropriate subdirectories in this model directory, and export the mesh files and textures from 3dsmax or gmax, into the directory. After exporting the mesh file, remember to use the Resource Collector from within 3dsmax or gmax to gather and place all the texture files in the directory.
7. Continue to add containers and tags in CCP for the model. Now you have a .im mesh file in the directory, additional information and choices will be available for data entry, for instance, selecting the mesh file will make it visible in the graphics window; when choosing attachment points within the mesh for tag data, you will be offered the actual mesh attachment point name for choosing.
8. Save the config.txt file (CCP knows where to place it, in the created directory). If any error messages appear, click with the Left Mouse Button (LMB click) on each error message to be taken to the tag within the model. Examine the data and fix the errors.
9. Some entries and changes to the asset may not

appear until the asset is committed. RMB click on the "New Asset" name in the main screen of CMP and select Edit/Commit. The temporary directory is closed and any new error messages that are generated may be examined.

10. Launching Trainz from within CMP will allow the new asset to be examined after placing in Surveyor. Of course if there are dependencies that have not been specified at this early stage, or not enough containers or tags added in CCP, the asset may not show up in Trainz.

11. Alternately, when working on the asset, the asset does not have to be committed manually to view it in Trainz. When launching Trainz you will be asked to confirm that any assets open for editing should be committed before proceeding. Any uncommitted asset will not show in Trainz.

12. If you have selected the settings within CMP to "re-open asset after exiting Trainz", the temporary directory will be re-opened automatically for you to continue working with the asset files.

13. It is important to backup assets, particularly as the exported files and textures are incorporated in the Trainz data base on committing the asset, and are not available for examination or copying once the temporary directory has been closed. Refer to [Page ..](#) for options to back up the asset files.

Each creator will determine a suitable workflow process with experience. CCP should make it easier for new creators to create workable config files.

Of course you can examine the saved config.txt file in the asset temporary directory, or even edit and re-save the file. This however is not the recommended process, as the config.txt file has then not been checked for errors in the CCP module, and could have newly introduced errors.

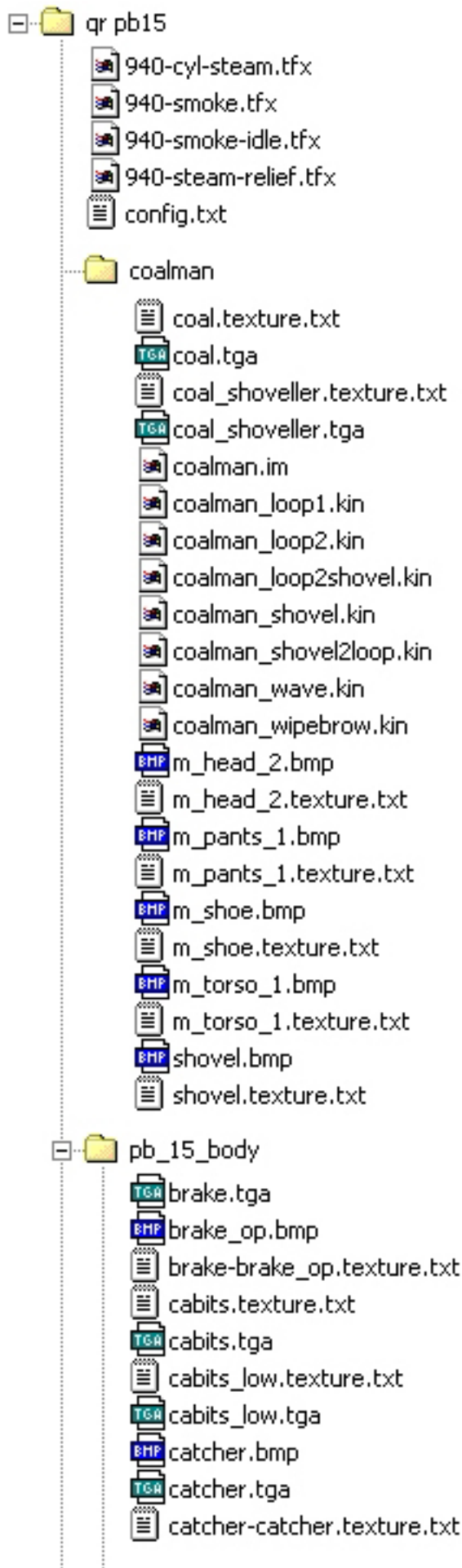
It is preferable that CCP be used to build a working config.txt file, and this process will validate the asset as suitable for upload to the Download Station.

Example Asset

The example asset chosen is the PB15 locomotive, which is included in Trainz. The various screenshots will illustrate how the graphical user interface would appear for this asset, and show the various forms of data value input. It will not discuss all the containers or tags in detail, but rather give a feel for the process of creating a new asset.

There are many containers and tags to make a fully functional locomotive asset, and duplicate containers and tags have not been shown in this example. Please refer to [Chapter 5](#) and [Chapter 6](#) for details on containers and tags for all asset kinds.

Example Asset PB15 Directory Layout



Main Directory: The directory structure for this example consists of a main directory “qr pb15”. This is also the name of the asset taken from the Username tag.

The .tfx files in this directory are Twinkles smoke effect files, created in the [Twinkles Editor](#) and placed in the directory by the creator.

The config.txt file created by CCP will also be placed in this top directory.

Coalman Directory: The asset has an animated coalman in the cab. For convenience, a subdirectory “coalman” has been created for the relevant files. This sub-directory includes the exported coalman.im mesh file, a number of animation files, coalman_loop1.kin file for example, and all the textures for the coalman model.

Main diffuse texture files (main color texture placed and mapped on the mesh in 3dsmax or gamax) in this directory are .tga type files.

The .bmp file format is used for opacity files associated with the texture files. These files allow certain parts of the main texture file to be transparent or translucent, depending on the amount and location of grey or black areas in the file. Opacity can also be included as an Alpha layer of the main .tga file, this is more efficient as Trainz only has to load one file instead of two separate files.

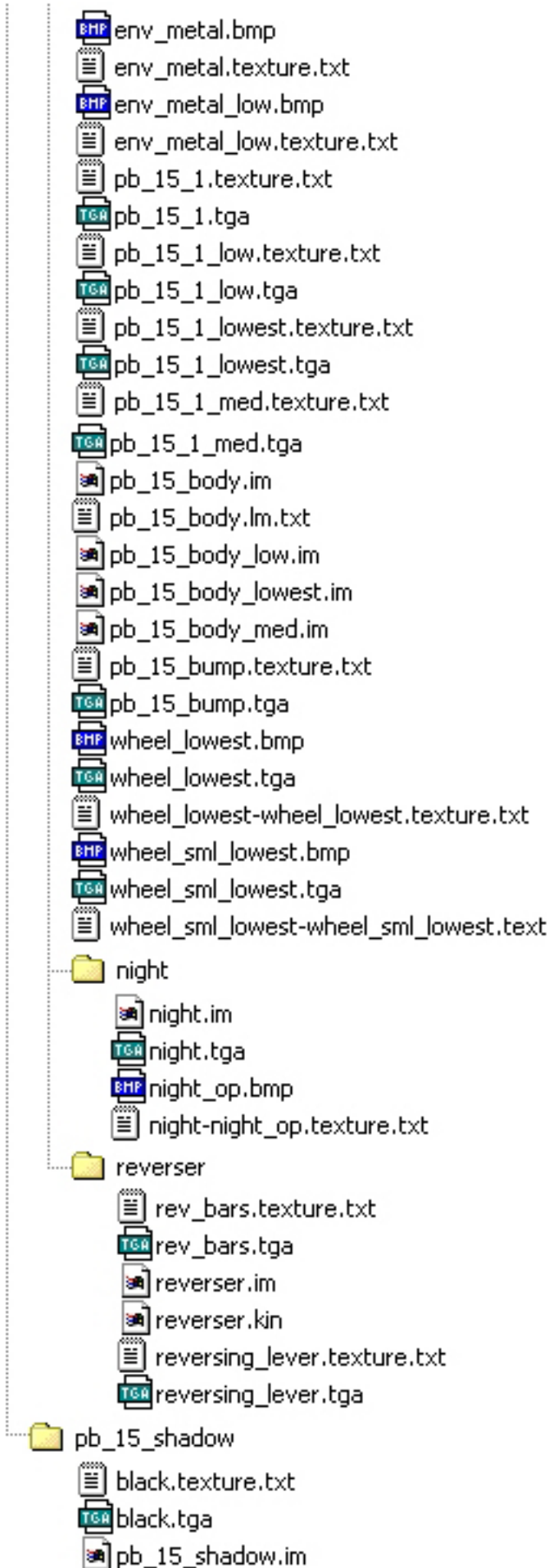
Please refer to [Page ...](#) for the correct file format and file sizes to be used for the model asset.

Main Body Directory: The main body mesh of the model is placed in the pb_15_body subdirectory. Files similar to the ones described in the coalman directory, but specific to the main locomotive body, are also included in this directory.

When the mesh file is exported from 3dsmax or gmax, a texture.txt file is automatically generated for each combination of textures, for example the catcher-catcher.texture.txt file. This text file includes details of the .tga diffuse texture and the .bmp opacity file that have been used in combination to texture the catcher part of the locomotive. The opacity file will make parts of the metal catcher on the front of the locomotive transparent, showing the distinctive steel bars in the typical steel catcher.

When making a complicated model, there may be other textures gathered in the Resource Collector process into this directory. These textures may have been for another mesh part to be exported later, and are not necessary for this part of the model. All textures without a matching texture.txt file should be removed from the directory.

Likewise, if texture file names are changed during the model development, unused texture.txt files may be left in the directory. If these files do not have the associated



texture files present, they can cause Trainz to crash.

Good practice at the end of the model development, is to remove all the files except the directory structure, the mesh.im files and config file, and re-export all meshes and the correct texture files. By leaving the mesh.im files, these can be overwritten during the export, without errors being introduced by re-typing the mesh names (they must match the mesh names iused in the config.txt file).

LOD files: There are some special mesh and text files in the main body sub-directory, to use LOD (Level of Detail) attributes to enhance frame rate in Trainz. The aim in Trainz is to keep models as simple as possible (reasonably sized textures, lowest polycount) to achieve a high frame rate (minimum display time between frames).

Briefly, Level of Detail (or 'LOD') is a technique used for asset mesh reduction. Trainz uses a different mesh dependant on the viewing distance, a lower quality meshes as viewing distance increases.

This concept is different from the previous "progressive mesh" (.pm) reduction. That is, instead of gradually reducing the polycount of a single mesh you can now have several versions of the same asset, each at different polycounts and texture levels.

Assets with LOD reduction must comprise of 'indexed meshes' or .im files only (exported from gmax or 3dsmax). No .pm files are used in LOD. TRS2004 looks for these .im files through an .im.txt (LOD mesh file) which is referenced from the asset's config.txt file.

The pb_15_body.im.txt file in this example is the text file that lists the details of the various meshes to be used, depending on viewing distance. The four referenced files are called pb_15_body_lowest.im, pb_15_body_low.im, pb_15_body_med.im and pb_15_body.im.

The last file is the highest quality for clos viewing distance.

Please refer to [Page ..](#) for details on Level of Detail

Night Directory: This is an example of the night mesh being placed in a night subdirectory.

Reverser Directory: The reverser.im mesh file is for the reverser used in the cab of the locomotive. As it is animated, an animation file, reverser.kin has also been created and exported from 3dsmax or gmax.

Shadow Directory: This is used to cast a shadow of the locomotive on the ground when the Shadow option is turned on in the Trainz configuration setup menu. This is a very simple low polygon mesh (to assist Trainz performance), to match the outline of the locomotive shape, and textured plain black.

Example Asset Main CCP Screen

This is the main CCP screen for the model, showing the dcontainer structure to the left and tags that have been entered in the top container, the Kind Traincar container.

The screenshot displays the Main CCP (Content Creation Panel) for a traincar asset in Trainz Railroad Simulator 2006. The interface is divided into two main sections: a left-hand dcontainer structure and a right-hand main configuration area.

Left-hand dcontainer structure:

- traincar
 - obsolete-table
 - mech-table
 - default
 - shadow
 - reverser
 - default-nick:forward
 - occamar
 - cncks0
 - cncks1
 - cncks2
 - cncks3
 - cncks4
 - cncks5
 - cncks6
 - cncks7
 - kuid-table
 - boogys
 - 0
 - 1
 - 2

Main configuration area (traincar):

kind [171456] [10124] [0]

category-class [4E - Electric Locomotive]

interior [PE_interior] [..] [..] [X]

engine [X]

name [OR PB E how]

mass [37000]

nick [4J] [X]

kind [traincar]

hms [0] [X]

enginespec [PE Engine curty] [..] [..] [X]

engineound [pb15] [..] [..] [X]

hornound [pb15] [..] [..] [X]

smoke_start [0.3] [X]

smoke_random [2] [X]

smoke_sound [1] [X]

smoke_lifetime [6] [X]

smoke_height [7] [X]

smoke_speed [4] [X]

description [The 366in class PE - E Class were built throughout the early 1900's by W.G. & Co. of Mayborough. The locomotives were of 4-6-2 wheel arrangement, and were a variation of the H-15 Class, the H-15 Class of the 19th Century. Early examples were fitted with mechanical valve gear, but those built from 1920 onward were fitted] [X]

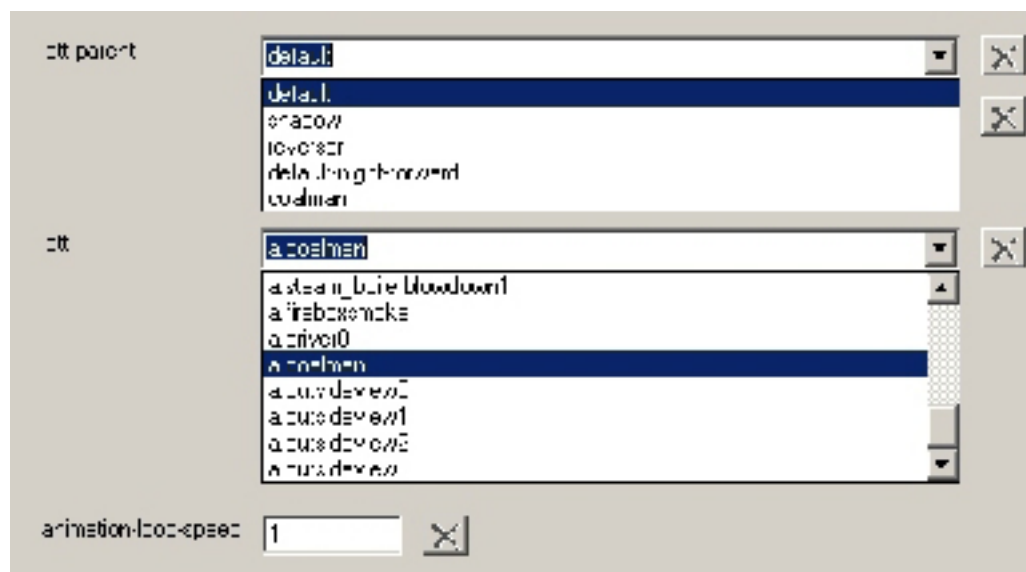
category-region [AL] [AM] [AN] [AO] [AP] [AQ] [AR] [AS] [AT] [AU] [X]

category-era [1825] [1835] [1845] [1855] [1865] [1875] [1885] [1895] [X]

threshold [2.5]

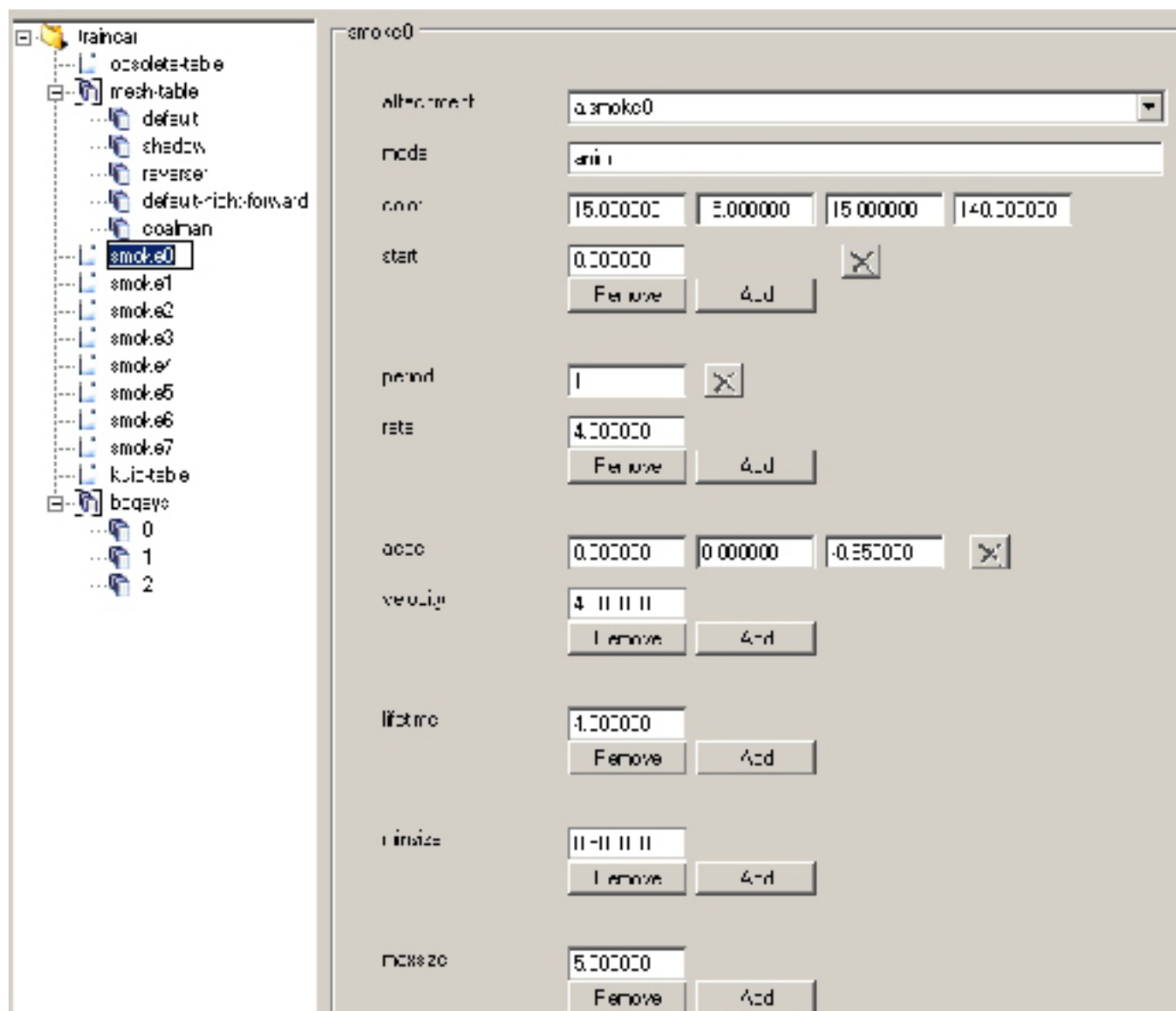
Example Asset Dropdown Selection Box for the Coalman Mesh

The selection box uses a dropdown menu, to display the attachment points found in the mesh model, for selection.



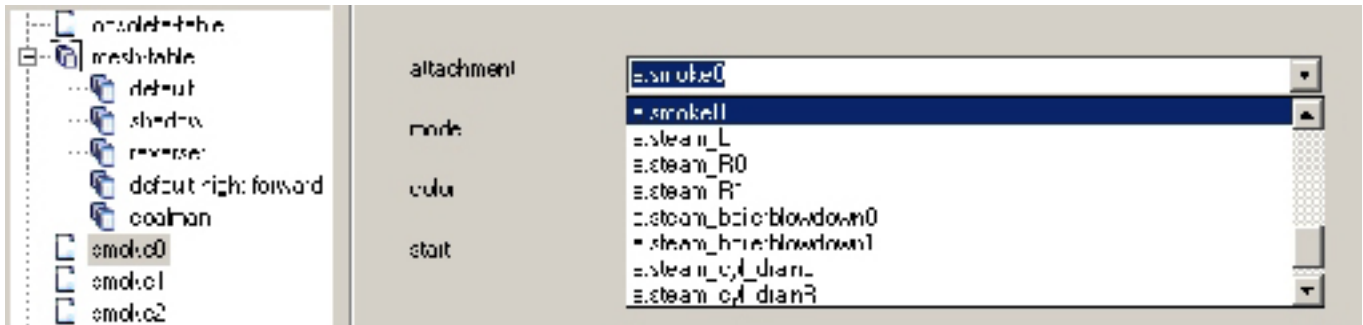
Example Asset Smoke Container

The menu allows appropriate values to be entered for the smoke effects. Note that some values can be removed or other input boxes added for additional values for the tag .



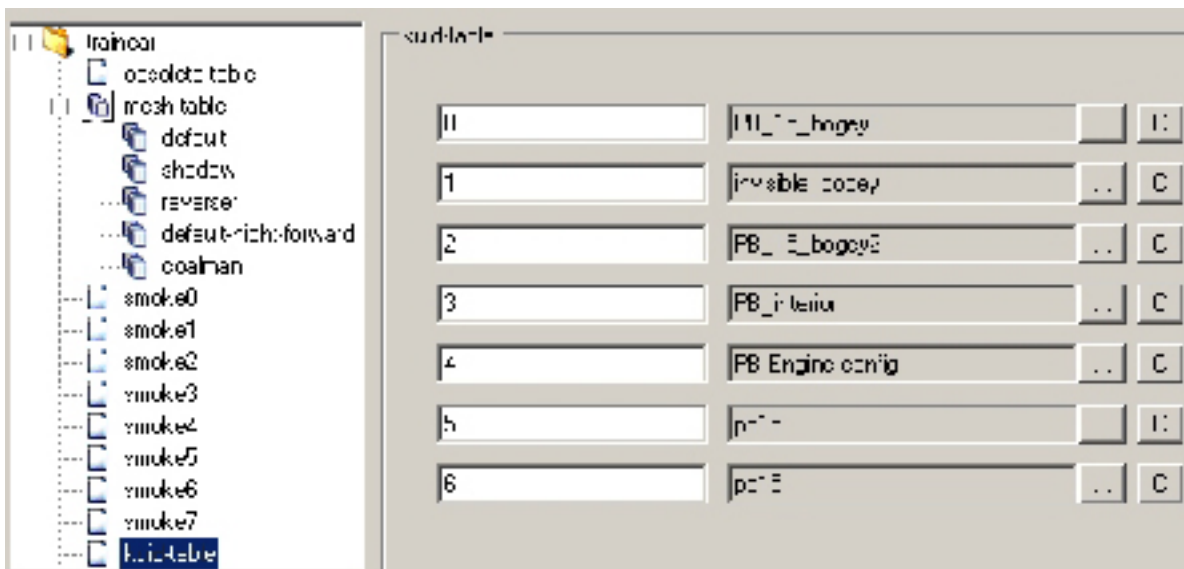
Example Asset Smoke Attachment Dropdown Box

The selection box uses a dropdown menu, to display the attachment points found in the mesh model, for selection.



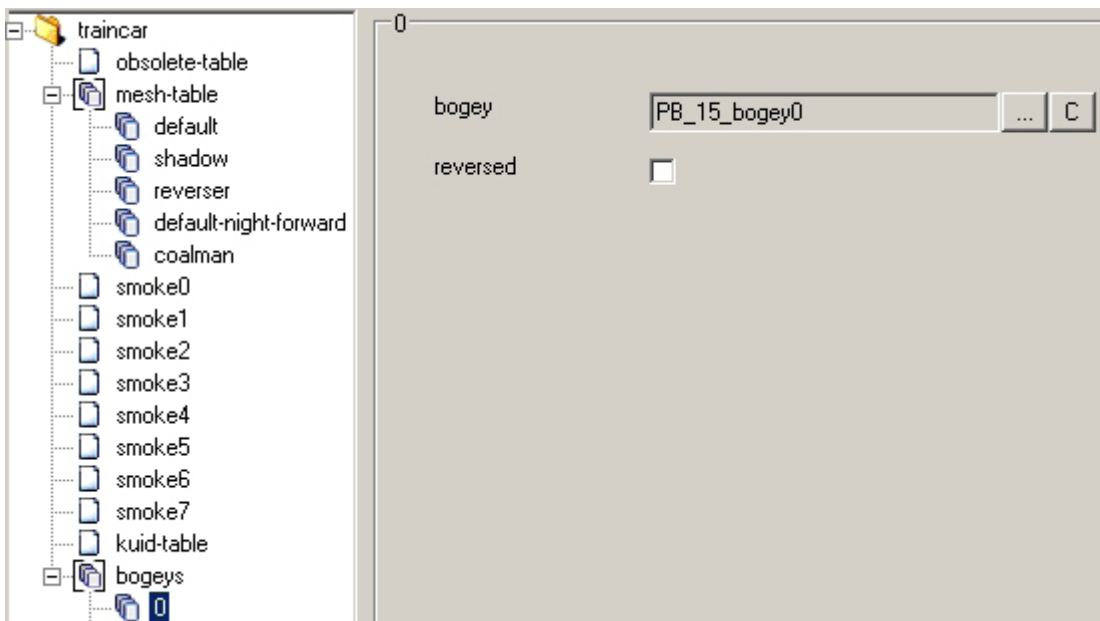
Example Asset Kuid Table Container

A browse box is provided to select the appropriate kuid for the table.



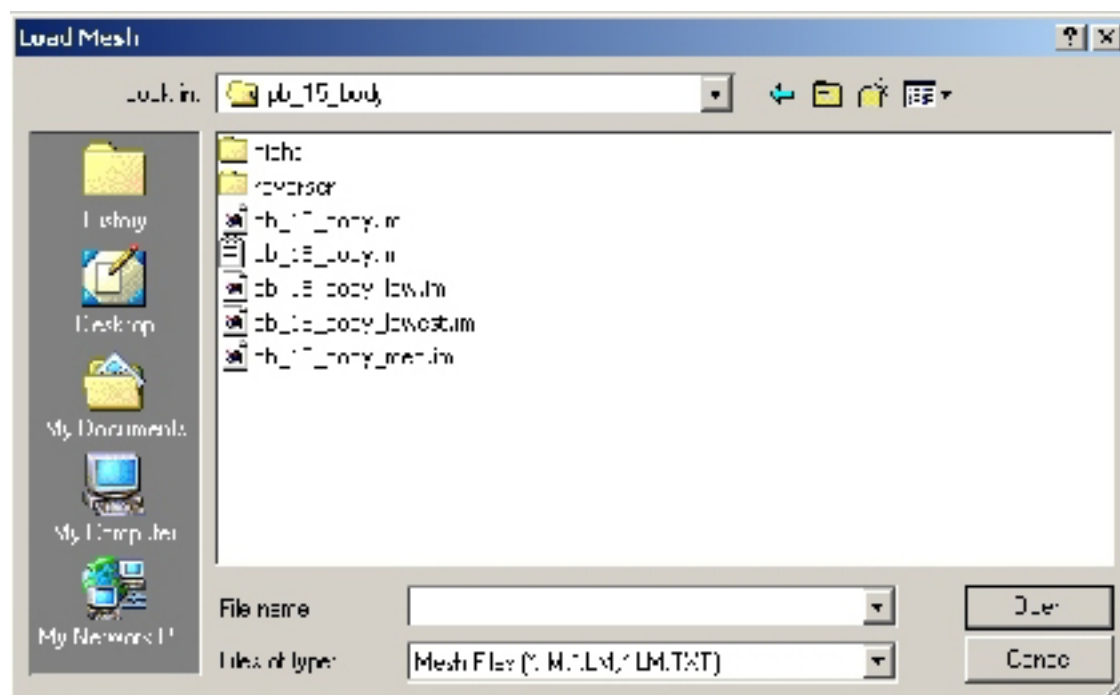
Example Asset Bogey Container

A browse box is provided to select the appropriate bogey for the model.



Example Obsolete Table and Mesh Table Browser

A kuid browse box presents the assets as proper names for selection, not kuids, and the mesh browser lists the available meshes in the model directory.



CHAPTER 5

Common Containers and Tags

The purpose of this chapter is to detail those containers and tags which are common across all kinds, and to detail their usage.

Chapter 5 Contents

Across all kinds, a number of tags and containers are required as basic prerequisites (eg. kuid tags), while others apply to all kinds though they aren't mandatory (eg. author information tags).

This section details those tags and containers which are present in all kinds, as well as some of the more regularly used containers/tags.

The following tags and containers are defined in this chapter. Note that additional notes on their usage are shown in Chapter 6 where relevant.

Common Containers

- Kuid Table
- Obsolete Table
- String Table cn, cz, de, es, fr, it, pl, ru
- Thumbnails
- Extensions

Common Tags

- kuid
- trainz-build
- category-class
- category-region
- category-era
- username cn, cz, de, es, fr, it, pl, ru
- kind
- description cn, cz, de, es, fr, it, pl, ru
- author
- organisation
- contact-email
- contact-website
- category-keywords
- license

Other Regularly Used Containers

Mesh Table

- anim
- auto-create
- animation-loop-speed

- critical-animation
- use-parent-bounds
- att
- att-parent
- opacity
- light
- test-collisions
- mesh
- night-mesh-base
- radius
- collision-parent

Effects

Name Effect

- name
- font
- fontsize
- fontcolor

Corona Effect

- kind
- att
- directional
- frequency
- max-distance
- object-size
- texture-kuid
- wave-shift

Texture Replacement Effect

- kind
- texture
- Attachment Effect
- kind
- att
- default-mesh
- surveyor-only

Animation Effect

- kind
- anim
- looped
- speed

Tracksound Container

- track-sound
- priority
- track
- track-parent
- bogey

Soundsript Container

- repeat-delay
- distance
- ambient
- attachment
- nostartdelay
- trigger
- value-range
- volume

- sound

Queue Container

- size
- animated-mesh
- custom-attachments
- initial-count
- passenger-queue
- product-kuid
- allowed products
- conflicts-with-queues
- attachment-points
- allowed-categories

Smoke Container

- attachment
- mode
- color
- rate
- velocity
- lifetime
- minsize
- maxsize
- accel
- conesize
- direction
- enabled
- endcolor
- faces
- file
- inherit-velocity
- interpolate
- loop
- loopdelay
- maxrate
- maxspeedkph
- minrate

- period

- scale
- shift
- start
- texture

Other Regularly Used Tags

- alias
- autaname
- class
- dighole
- floating
- height
- height-range
- icon0, icon1, icon2, icon3
- icont-texture
- light
- preview-mesh-kuid
- preview-scale
- nightmode
- rgb
- rollstep
- rotate
- rotate-yz-range
- rotstep
- script
- snapgrid
- snapmode
- surveyor-name-label
- surveyor-only
- texture

Common Containers

The following containers are present in all kinds.

Kuid Table

A list of KUIDs required for this asset to function correctly.

A kuid-table must be included where the config references additional KUIDs, such as a bogey, or a pantograph. The Download Station performs a search, and those found are added to the download pack.

Obsolete Table

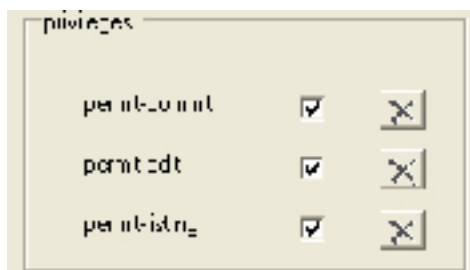
The obsolete-table describes the assets revision history.

This container was used extensively for pre-TRS2004 assets as each version required a unique Content ID. In order to simplify this process the KUID2 format was introduced, which now supersedes the obsolete-table method. TRS2006 and the Download Station automatically detect and use the most recent version of an asset whether it be through the KUID2 system or through the obsolete-table.

The obsolete-table container has been included to maintain backwards compatibility with older assets and it is recommended that the KUID2 system be used instead.

Privileges

As of TRS2006, limited content protection applies only to built-in (JArchived) assets. The following tags are available:



The follow permit tags grant or deny the user specific access rights. By default, all permissions are granted on an asset. Set one of these tags to false (0) to remove the permission.

permit-commit - Allows the end-user to commit changes to this asset.

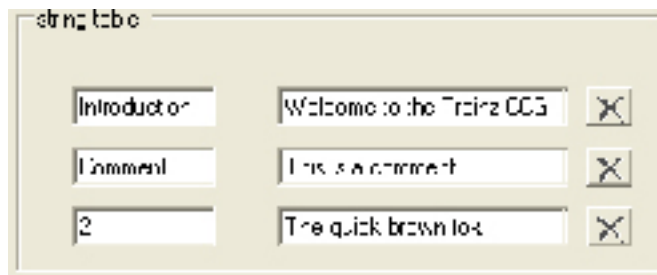
permit-edit - Allows the end-user to open this asset for editing.

permit-listing - Allows the end-user to view this object in the surveyor pickers (if it is of an appropriate kind.) Does not affect the visibility of the asset within the CMP asset list.

String Table cn, cz, de, es, fr, it, pl, ru

Every asset can have a string table. A string table is a list of text strings that are defined in the string-table section

of the asset's config.txt file as follows:



On their own, these string tables are useless. It's only when you get into scripting that the string tables become useful.

To compliment the english String Table, a variety of additional String Tables allow for equivalent strings to be supported across many languages.

Thumbnails

Any asset may specify a thumbnail or preview image. The exact usage of this image may vary depending on the asset kind and the build of Trainz, but the following usages are historically common:

32x32 - standard icon representation for display in lists, etc.

128x64 - 'kind traincar' list icon.

512x512 - 'kind traincar' preview image.

240x180 - Download Station thumbnail image.

To allow for the generic specification of thumbnail images, the following format is adopted for TRS2006.



Any number of thumbnail entries may be present. It is recommended that no more than 3 images are used. The specified width and height must match the actual width and height of the image file in pixels.

Supported image formats include:

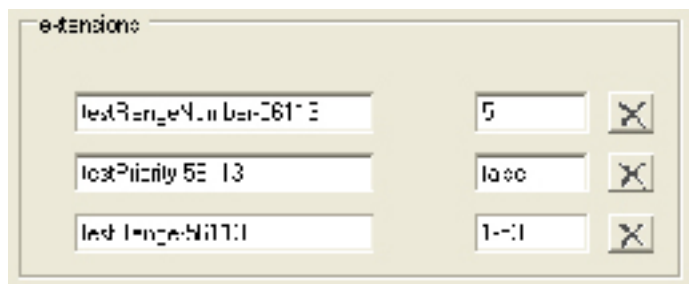
- 32-bit uncompressed targa (.tga) - this supports an alpha channel
- 24-bit windows bitmap (.bmp)
- Jpeg (.jpg)

When displaying an image, the closest-sized image (in terms of pixel dimensions) will generally be used.

Extensions

Third parties may sometimes wish to include additional config tags in an asset's config file for the purposes of providing asset-keyed data to custom scripts.

It is important that the following mechanism is used to prevent potential conflict with future Auran tags or other content creators.



All third-party tags must be embedded under an extensions container which is placed at the top level the config file.

Tags within the extensions container should have a meaningful name and should be end with a hyphen and the UserID of the content creator who is responsible for the extension. The responsible content creator should determine any rules and restrictions which apply to his or her tag and should make an effort to provide this information to other creators (outside the scope of the asset itself).

While creators are permitted to make use of each other's extensions, it is not permissable to create a new extension (or change the meaning of an extension) in the namespace belonging to another creator.

Each tag in the extensions container may be either a single value, or a subcontainer. If a subcontainer is used, the contents of the subcontainer should be specified in the extension's documentation.

No restrictions are placed by Auran on the values within a specific extension, beyond the normal config file format guidelines.

Common Tags

The following tags are present in all kinds.

kuid

The "unique identifier" number for the asset. When creating new assets with Content Creator Plus, the asset kuid will be automatically generated.

See Chapter 1 for more information on kuids.

Example value: "56113:1107:0"

trainz-build

The Trainz build number for which the asset was created. TRS2006 assets will have a trainz-build number of 2.5.

TRS2006	2.5
TRS2004-SP4	2.4
TRS2004-SP3	2.3
TRS2004-SP2	2.2
TRS2004-SP1	2.1
TRS2004	2.0
UTC	1.5
Trainz SP3	1.3
Trainz 1.2	1.2
Trainz 1.1.1	1.11
Trainz 1.1	1.1
Trainz CE	1.0

NOTE: When editing an asset in Content Creator Plus, the trainz-build will automatically be upgraded to 2.5 in order to support newly added tags and functions.

Example Value: "2.5"

category-class

The class code for this asset. A full list of class codes is provided in an appendix at the end of this document.

category-class has been in use since Trainz SP3.

Example Value: "LM - Monorail Vehicles"

category-region

The region code for this asset. A full list of region codes is provided in an appendix at the end of this document.

Similar in function to the category-region-XX tags which have been in use since Trainz SP3, however category-region has been reformatted to allow multiple regions on a single line.

A semicolon is used to separate individual values. Whitespace is not permitted.

Example Value: "AD;AE;AU"

category-era

The era code for this asset. A full list of era codes is provided in an appendix at the end of this document.

Similar in function to the category-era-XX tags which have been in use since Trainz SP3, however category-era has been reformatted to allow multiple eras on a single line.

A semicolon is used to separate individual values. Whitespace is not permitted.

Example Value: "1800s"

username cn, cz, de, es, fr, it, pl, ru

The human-readable name of the asset. The username tag is mandatory. To compliment the english username, a variety of additional optional tags allow for equivalent usernames to be supported across many languages. These are as follows:

username-cn - Chinese
username-cz - Czech
username-de - German
username-es - Spanish
username-fr - French
username-it - Italian
username-pl - Polish
username-ru - Russian

Example Value: "testActivity"

kind

The asset kind. Must be one of the Auran-supplied asset kinds. I.e. kind industry

See Chapter 2 for a complete list of kinds.

Example Value: "MOSignal"

description cn, cz, de, es, fr, it, pl, ru

The human readable description of the asset. This is an optional tag. To compliment the english description, a variety of additional tags allow for equivalent descriptions to be supported across many languages. These are as follows:

description-cn - Chinese
description-cz - Czech
description-de - German
description-es - Spanish
description-fr - French
description-it - Italian
description-pl - Polish
description-ru - Russian

Example Value: "This is a test activity written for the 2006 CCG."

author

The human-readable author name text. This is an optional tag.

Example Value: "Scott Cameron"

organisation

The organisation name as specified by the asset author. This is an optional tag.

Example Value: "Auran"

contact-email

Email address specified by the asset author. This is an optional tag.

Example Value: "helpdesk@auran.com"

contact-website

Website URL specified by the asset author. This is an optional tag.

Example Value: "www.auran.com"

category-keywords

To improve the searchability of assets, generic search keywords may be included in the config file.

Each keyword should be a lowercase word with no punctuation. Keywords should be English and as neutral as possible.

Keywords are separated by a semicolon. Whitespace is not permitted within the category-keywords tag.

Example Value: "scott;auran;example;ccg;test"

license

The asset's license agreement as written by the author. This is an optional tag.

Example Value "This is an example license"

Other Regularly Used Containers

Some containers are not present in every single kind, but appear across a number of different common kinds. The most common of these are detailed in this section.

Mesh Table

This is the preferred method of asset mesh placement for most mesh asset types. It gives huge control over mesh placement and animations.

There are some asset types that cannot use a Mesh Table. These include all Bridges, Tunnels, Rails, Pantographs and other Spline Objects (eg. Fences or Catenaries).

Important Note: Any asset that uses a mesh-table will not be compatible with pre-TRS 2004 versions of Trainz (I.e. Ultimate Trainz Collection or UTC). TRS will of course still read UTC assets. Just remember that as with most major software releases, backwards compatibility is usually achievable, while forwards compatibility is often impossible.

Mesh Tables contain the following tags and subcontainers:

anim: The animation file (.kin) exported from 3d Studio Max or Gmax. This may include a sub-path. [More information on .kin files is available in Chapter 8 of this](#)

[document](#).

auto-create: Generated automatically when placed. In some instances you don't want the mesh visible (as this may be controlled through script). If auto-create is 0 the mesh will not be visible when placed.

animation-loop-speed: This tag must be here if the asset is to animate when placed. If this tag is not present when placed the animation will not play by default, but may play if controlled by script.

critical-animation: When enabled, this forces the animation to continue playing when off screen. Impacts performance when enabled.

use-parent-bounds: Specifies that the mesh should use the bounds of the parent object for visibility culling. Use with caution.

att: The mesh (and animation if present) is inserted at a mesh attachment point rather than the origin.

att-parent: The insertion attachment point is located within the mesh from the 'name' block in the config.txt.

opacity: Controls the opacity of the mesh. Zero (invisible, not recommended) or one (solid).

light: Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is light by general light value, 1 sets directional light which is affected by the position of the sun.

test-collisions: This is an interior-specific mesh-table tag. When disabled, it prevents the mesh from obstructing the mouse (eg. if a mesh overlays a lever and should not be tested). Enabled by default.

mesh: The 'main' mesh name. This may include a sub-path. ie: mesh "nightwindows/nightwindows.im".

night-mesh-base: This night mesh is linked to the default mesh and is visible only when the 'default' mesh is visible.

radius: radius for notches display.

collision-parent: For collision-proxy meshes in an interior mesh-table, this specifies the parent object to be proxied.

Effects (optional mesh variables)

The effects containers are a subset of the Mesh Table Container. At the time of writing there are 4 distinct types of effects which are:

Name Effect, Corona Effect, Attachment Effect, and Animation Effect.

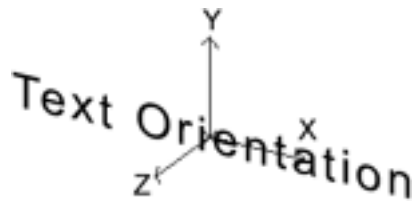
Name Effect

Some assets may have editable signs. When you set an asset's name in surveyor through the Edit Properties icon ('?' icon) the signage can be set-up to automatically

update. The variables can be set for each sign.

kind: The effect kind.

att: The Sign Text insertion point (part of the mesh). See below for correct orientation:



name: The default text when placed. If not used it will default to the config 'block' name and will not be editable. When name is specified, it uses the asset's changeable name.

font: The name of the font. Generally shouldn't be used as there is no guarantee that the font exists on an end-users system.

fontsize: The size of the sign text.

fontcolor: The colour of the sign text in r.g.b.

Corona Effect

A corona is a 'glow' light effect. It is a simple texture that is inserted at an attachment point within the mesh. Corona's can be added to any asset that uses a mesh-table.

Examples of coronas used in-game can be seen on the "Airport" and "Airport Basic" assets. The Jumbo Jet, the Cessna and the Airport tower all use flashing corona's.

kind: The effect kind.

att: The corona insertion point and centre (part of the mesh).

directional: The default for coronas is to be aligned to the attachment point to face the NEGATIVE Z direction. This is especially useful for Trainscars.

frequency: This variable specifies the frequency in Hz (or 'flashes' per second). Eg. 1 for once per second, 0.5 for once every 2 seconds, 2 for twice in a second.

max-distance: Maximum distance to which the effect is visible.

object-size: Size of the corona on the object when viewed up close. Defaults to 0.15 (ie . 0.15m).

texture-kuid: Add this tag only when you want to specify your own texture for the corona. It specifies the KUID of a kind texture asset. If the texture-kuid tag is not present the corona will use the default yellow/orange texture in TRS.

wave-shift: Affects the flashing intensity pattern on the corona.

Texture Replacement Effect

This effect was created for rollingstock items to swap the visible texture of bulk loads (such as coal or woodchips).

If a coal car is set up to take any bulk load (which includes woodchips) the 'coal' texture on the load mesh will update to a 'woodchips' texture when it loads woodchips.

kind: The effect kind.

texture: The replacement texture.

Attachment Effect

In TRS we now have the ability to attach a mesh into another mesh by referencing it's kuid through a mesh table.

Prime examples are the TRS fixed-track assets. Rather than having an arrow in each fixed-track asset directory, we saved a lot of memory space by making it reference a KUID of that mesh. Therefore it only needed to be cached once. Use attached meshes should be only of kind scenery or kind mesh.

WARNING: Never cross-reference a kind attachment kuid with the assets own kuid. Unless of course you want to show your friends what an instant fatal error looks like!

kind: The effect kind.

att: The insertion point of the attached mesh. In the example left, it is the insertion point of the 'default' mesh.

default-mesh: The KUID of the attached mesh.

surveyor-only: Adding this means the attached mesh will only be visible in Surveyor and not Driver.

Animation Effect

This effect is used when a mesh has a variety of animations. Usually the animations will be controlled by a script related to the asset.

An example of the kind animation effect is the PB15 interior Coalman. The script for this ties in the animations with the coal requirements of the steam locomotive.

kind: The effect kind.

anim: Reference to the animation file (.kin).

looped: Use only if the animation is looping. Default 0 (i.e. not looped).

speed: Speed factor of the animation. Default 1. 2 = Double speed.

Kinds that use the Mesh Table Container: *bogey, buildable, drivercharacter, fixedtrack, industry, interior, mesh, mocrossing, mojunction, mosignal, mospeedboard, pantograph, product, scenery-trackside, scenery, traincar, turntable.*

Tracksound Container

The tracksound container stores information regarding custom tracksounds that can be attached to certain assets.

The tracksound container contains the following tags:

track-sound: The kuid of the tracksound object to be used.

priority: The priority of the sound versus other sounds to be played. Lower values indicate a higher priority.

track: The track type to which this sound will apply

track-parent: The parent (eg. bridge/industry/tunnel) of the track to which this sound will apply.

bogey: The bogey to which this sound will apply.

Kinds that use the Tracksound Container: *bogey, bridge, chunky-track, double-track, mesh-reducing-track, track, tunnel.*

Soundscript Container

Soundscripts give ambient or directional sounds to objects. They cannot be used on track, bridge or spline objects. Wav files should be located within the same directory as the config.txt file.

The Soundscript Container contains the following tags and sub-containers:

repeat-delay: 1 or 2 numbers (min, max, in seconds) time to delay between the end of the sound playing, and playing it again randomised between (min .. max) default min = 0, default max = min distance.

distance: Two numbers (in meters). The first number is the distance at which the sound is played at 100%. The second number is the he cut-off distance. Doesn't affect the volume of the sound default: 50m, 150m.

ambient: Ambient sounds have no 3d "position" and may be stereo. non-ambient (positional) sounds are positioned on the object and must be mono.

attachment: Attachment point on the object to attach the sound to.

nostartdelay: If not set, the sound will have a short delay before playing. This stops flanging (a really nasty sound caused when several copies of the same sound are

played at once).

trigger: Currently used only for levers. the sound doesn't play until the trigger message happens.

value-range: Two numbers, currently used only for day/night sound effects. Midnight is 0.5, midday = 0.0 or 1.0. Where the numbers are not the same, this sets the start and end times for the sound to play. Default 0,0 (off).

volume: Gain of the sound. Default 1.0 = 100%.

Sound: Filename of the sound to be played.

Kinds that use the Soundscript Container: *buildable, fixedtrack, industry, interior, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

Queue Container

The queues container states which product or products the industry can use. It contains the size of each product, the initial count when placed, and can refer to it's visual load state whether through a load animation or attachment.

Any load animations are set-up within the mesh-table.

The queue container contains the following tags:

size: Size of queue.

animated-mesh: Animated mesh which changes as the queue becomes full.

custom-attachments: Not used.

initial-count: The initial number of items in the queue.

passenger-queue: Not used.

product-kuid: The product type used to fill 'initial-count'

allowed products: The allowed products in this queue.

conflicts-with-queues: This queue and the conflicting queue(s) cannot be used simultaneously.

attachment-points: List of attachment points for this queue on which products are visualised. (Use this, OR animated-mesh)

allowed-categories: The allowed product categories in this queue.

Kinds that use the Queue Container: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

Smoke Container

More information on smoke and particle effects can be found in [Chapter 11](#).

The Smoke Container contains the following tags:

attachment: The attachment point (stored in the mesh file) to place the smoke effect.

mode: Describes the mode or type of this smoke effect. This affects how start and period are interpreted. (time | speed | anim | timeofday). [See Chapter 11](#) for a more detailed description of the mode tag.

color: The A,R,G,B colour value of the effect.

rate: The rate of emission in particles per second for modes time, speed, and timeofday, or the number of particles to emit over the animation period for anim mode. Default is 4.

velocity: The initial speed of emitted smoke particles. Default is 1.

lifetime: Time in seconds that smoke particles exist for. Default is 3.

minsize: Start size of smoke particles. Default is 0.

maxsize: End size of smoke particles. Default is 3.

accel: Acceleration. A vector pointing in the direction of the sum of all forces affecting this smoke effect. Essentially, <z> describes gravity, and <x>, <y> describe the force of wind. Default is 0,0,0.

conesize<-----TEXT TO GO HERE----->

direction: The vector at which the smoke travels.

enabled: Specifies whether the effect is enabled.

endcolor: The final colour the smoke effect shifts to.

faces: The direction the smoke effect faces. (camera, motion, down)

file: The twinkle file to be used (optional).

inherit-velocity: blah<-----TEXT TO GO HERE----->

interpolate: blah<-----TEXT TO GO HERE----->

loop: Time in seconds to loop the smoke sequence. Only valid if mode is set to time.

loopdelay: Delay (in seconds) before the effect is played again.

maxrate: The maximum rate at which particles are emitted.

maxspeedkph: blah<-----TEXT TO GO HERE----->

minrate: The minimum rate at which particles are emitted.

period: The usage of period depends on the value of the mode tag.

If the mode is set to time, period is the duration of time this effect will remain active.

If mode is set to anim, period is a value from 0.0 to 1.0 that describes the duration over which the effect is active. start + period must not exceed 1.0.

In all modes, period can be set to -1 (default) to imply the phase is active until the next phase begins.

scale: blah<-----TEXT TO GO HERE----->

shift: blah<-----TEXT TO GO HERE----->

start: The usage of start depends on the value of the mode tag.

If the mode is set to time, start is a set of time values in seconds after the creation of this effect's parent object when this phase of the effect will start.

If the mode is set to speed, start is a speed in meters per second (m/s) and period is not used. (Note: 1 m/s = 3.6 km/hr.) All other sequence attributes (rate, velocity, lifetime, minsize, maxsize) are interpolated so there are smooth transitions between phases.

If the mode is set to anim, start is a value from 0.0 to 1.0 which describes the start time into the object's animation cycle.

If the mode is set to timeofday, start is a value from 0.0 to 1.0 which describes the time of day when this effect will start.

Values range as follow:

0 - midnight, 0.25 - 6am, 0.5 - midday, 0.75 6pm, 1.0 - midnight.

texture: Kuid of the texture to be used for the effect.

Kinds that use the Smoke Container: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

Other Regularly Used Tags

Some tags are not present in every single kind, but appear across a number of different common kinds. The most common of these are detailed in this section.

alias

Kuid of the asset to be referenced as a basis for the new asset. For example TRS Traincars can reference archived locomotive mesh assets for use with custom textures. This process is done by aliasing the KUID of the archived traincars.

Kinds that use this tag: *bogey, bridge, buildable, chunky-track, double-track, drivercharacter, fixedtrack, industry, interior, mesh-reducing-track, mesh, mocrossing, mojunction, mosignal, mospeedboard, pantograph, product, scenery-trackside, scenery, track, traincar, tunnel, turntable.*

autoname

When enabled, automatically assigns a unique name to this object as it is placed.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

class

This refers to the name of the script file and the class of asset it is (the class must match that stated within the script file).

Kinds that use this tag: *activity, behavior, bogey, bridge, buildable, chunky, track, double-track, drivercharacter, drivercommand, engine, enginesound, environment, fixedtrack, groundtexture, hornsound, html-asset, industry, interior, library, mesh-reducing-track, mesh, mocrossing, mojunction, mospeedboard, paintshed-skin, paintshed-template, pantograph, product-category, product, profile, region, scenery-trackside, scenery, steam-engine, texture-group, texture, track, tracksound, traincar, tunnel, turntable, water2.*

dighole

Specifies the number of grid segments (length, width) to be removed from the surveyor grid to accommodate the turntable pit.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

floating

Obsolete.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

height

Height from the track level to the base. Should be negative.

Kinds that use this tag: *bridge, buildable, double-track, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, tunnel, turntable.*

height-range

min, max. eg: height-range -10, 100. Where min and max are values in meters. This allows you to specify the minimum and maximum height ranges for when adjusting the height of this object with the new "Adjust Height" tool in Surveyor's 'Object Tools' panel.

All scenery objects have a default min/max height range of 0 and 0. ie they do not (by default) allow you to adjust the height. Adding a height range is particularly useful for ships/buoys (placed on water) and for Station accessories.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

icon0, icon1, icon2, icon3

Small icon displayed over the vehicle preview in Surveyor.

Kinds that use this tag: *bogey, buildable, drivercharacter, fixedtrack, industry, interior, mesh, mocrossing, mojunction, mosignal, mospeedboard, pantograph, product, scenery-trackside, scenery, traincar, turntable.*

icon-texture

Now Obsolete. Thumbnails container should be used instead.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, product, scenery-trackside, scenery, traincar, turntable.*

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is light by general light value, 1 sets directional light which is affected by the position of the sun.

Kinds that use this tag: *bridge, buildable, chunky-track, double-track, fixedtrack, industry, mesh-reducing-track, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, track, traincar, tunnel, turntable.*

preview-mesh-kuid

Only add this to reference a different mesh for the Surveyor preview window. This is useful when an asset has a large bounding box. Ie the "Airport" with it's jet animation.

Kinds that use this tag: *bogey, buildable, drivercharacter, fixedtrack, industry, interior, mesh, mocrossing, mojunction, mosignal, mospeedboard, pantograph, product, scenery-trackside, scenery, traincar, turntable.*

Preview-scale

Scale of the preview mesh.

Kinds that use this tag: *bogey, buildable, drivercharacter, fixedtrack, industry, interior, mesh, mocrossing, mojunction, mosignal, mospeedboard, pantograph, product, scenery-trackside, scenery, traincar, turntable.*

nightmode

Only add this tag if you reference a default-night mesh in the mesh-table.

Values: home, lamp or constant.

Home switches on night effect at dusk and off sometime during the night.

Lamp switches the night effect on from dusk to dawn.

Constant lights are on day and night.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

rgb

Color used for display in mini map.

Kinds that use this tag: *bridge, buildable, chunky-track, double-track, fixedtrack, industry, mesh-reducing-track, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, track, traincar, tunnel, turntable.*

rollstep

Used in conjunction with rotate-yz-range, rollstep lets you specify the step size of roll angles (in degrees) for this object. Other example values are 1, 5, 20 etc. The default rollstep is 1.0.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

rotate

This lets you disable rotation on a scenery object. 0 to disable 1 to enable (default).

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

rotate-yz-range

Where min and max are values in degrees. This tag lets you set the roll / yz rotation range (normal object rotation is an xy rotation).

If you want your scenery object to support rolling then use this tag to set the minimum and maximum roll range. By default, objects have a min/max roll range of 0 to 0.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

rotstep

This lets you specify the step size of rotation angles (in degrees) for this object. Other example values are 1, 10, 20, 90, 180 etc. The default rotstep is 1.0

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

script

The script file (gs or gse file).

trackside, scenery, texture-group, texture, traincar, turntable.

Kinds that use this tag: *activity, behavior, bogey, bridge, buildable, chunky, track, double-track, drivercharacter, drivercommand, engine, enginesound, environment, fixedtrack, groundtexture, hornsound, html-asset, industry, interior, library, mesh-reducing-track, mesh, mocrossing, mojunction, mospeedboard, paintshed-skin, paintshed-template, pantograph, product-category, product, profile, region, scenery-trackside, scenery, steam-engine, texture-group, texture, track, tracksound, traincar, tunnel, turntable, water2.*

snapgrid

This lets you specify the size of the grid (in meters) the object snaps to.

We recommend factors/fractions of 720 as this is the size of a base board and the positioning may do odd things across section borders.

eg 1, 2, 5, 10, 20, 30, 40, 45, 60, 80, 90, 120, 180, 240, 360, 720.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

snapmode

Specifies the alignment of the turntable to the surveyor grid.

1 = origin snaps to grid (use for removing even dighole values), 2 = origin snaps to the center of a grid square (use for odd dighole values).

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

surveyor-name-label

Specifies whether this item has floating name label text.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

surveyor-only

Adding this means the attached mesh will only be visible in Surveyor and not Driver.

Kinds that use this tag: *buildable, fixedtrack, industry, mocrossing, mojunction, mosignal, mospeedboard, scenery-trackside, scenery, traincar, turntable.*

texture

An image texture file.

Kinds that use this tag: *buildable, drivercharacter, fixedtrack, groundtexture, industry, mocrossing, mojunction, mosignal, mospeedboard, product, scenery-*

CHAPTER 6

All Other Containers and Tags

Chapter 6 described the Containers and Tags that are common to all model assets. The purpose of this chapter is to define and describe the remaining Containers and Tags used in TRS2006, and to give examples of how they are used and entered in Content Creator Plus. Please refer to [Chapter 7](#) for example assets using the Container and Tags, and for the directory structure of model asset.

INTRODUCTION

In each section, an example config.txt layout is shown illustrating which values would be present in a basic asset of that KIND. These values are color coded to help represent their purpose:

RED - A default container. These containers only appear once in a config and cannot be renamed.

BLUE - A user defined container. These containers may appear multiple times as needed, and can often be renamed.

ITALIC - Italic text represents a data value. Three distinctions are made:

kuid - The value is a kuid number, which may be used to reference another asset

data - The usable data value for the tag is stored within the file

file - The value is a filename (and the file is located within the asset)

The following KINDS are outlined in this Chapter:

• Activity	• HTML-Asset	• Product-Category
• Behavior	• Industry	• Profile
• Bogey	• Interior	• Region
• Bridge	• Library	• Scenery
• Buildable	• Mesh	• Scenery-Trackside
• Chunky-Track	• Mesh-Reducing-Track	• Steam-Engine
• DriverCharacter	• MOCrossing	• Texture
• DriverCommand	• MOJunction	• Texture-Group
• Double-Track	• MOSignal	• Track
• Engine	• MOSpeedboard	• TrackSound
• EngineSound	• Paintshed-Skin	• TrainCar
• Environment	• Paintshed-Template	• Tunnel
• Fixed Track	• Pantograph	• Turntable
• GroundTexture	• Product	• Water2
• Hornsound		

KIND: ACTIVITY

Description

An activity is a scripted scenario that details the locomotives and rolling-stock used in a map, the driver settings, commands and scripts.

A train driver can undertake a sequence of planned moves – a scenario.

Container Structure

A well formed activity kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

activity

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
class	<i>data</i>
script	<i>file</i>

driver-settings

autopilotmode	<i>data</i>
startingtime	<i>data</i>
timerate	<i>data</i>
deraillevel	<i>data</i>
showhelp	<i>data</i>
controlmethod	<i>data</i>
weather	<i>data</i>
changeability	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “YS - Scenario”.

See the “Maps & Scenarios” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “activity”.

Class

The name of the scenario class within the script file.

Script

The script file (gs or gse file).

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the activity kind also contains additional containers and tags that are specialised to the requirements of the kind.

driver-settings

Specify the settings of this scenario, similar to Driver’s settings’ screen.

The driver settings container contains the following child tags:

autopilotmode

AI driver setting. (off, on)

startingtime

Time of day. Range is from 0 to 1 (0.5 - midday).

timerate

Time progression. (1 - real-time, 2 - double speed etc.)

deraillevel

Derail setting. (none, arcade, realistic)

showhelp

Show Driver Help. (off, on)

controlmethod

Driver control setting. (dcc, cabin)

weather

Weather setting. (clear, cloudy, drizzle, rain, stormy, light snow, medium snow, heavy snow)

changeability

Propensity for weather to change. (none, periodic, extreme)

Additional Tags

username

Name of scenario displayed in TRS.

scriptlibrary

Obsolete. Now replaced with the “script” tag.

scriptclass

Obsolete. Replaced with the “class” tag.

Notes

Scripted scenarios are made available for backwards compatibility reasons.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Driver Settings Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru

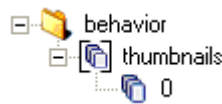
KIND: BEHAVIOR

Description

A configurable behavior module that forms part of a session.

Container Structure

A well formed behavior kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

behavior

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
class	data
script	file

thumbnails

0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “YR - Rule”.

See the “Maps & Scenarios” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “behavior”.

Class

The name of the scenario class within the script file.

Script

The script file (gs or gse file).

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

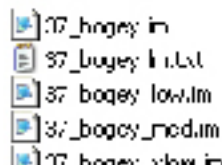
KIND: BOGEY

Description

Bogeys are locomotive or rolling stock wheel mechanisms, sometimes known as ‘Trucks’. This asset is for attachment to a “traincar” (locomotive or rolling-stock) and can include animation and a shadow model.

Container Structure

A well formed bogey kind has the following container structure:



See Chapter 7 for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

bogey

kuid	kuid
trainz-build	data
category-class	data
username	data
kind	data
category-region	data
category-era	data

mesh-table

default

mesh	file
auto-create	data

thumbnails

0

image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ZB - Bogie/Truck”.

See the “Track” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

kind

Must be “bogey”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Tags

animdist

Leave this tag out if the bogey is not animated.

The distance traveled in meters by the bogeys in 1 second (30 frames) of animation.

Bogey animations (exported from Gmax or 3ds Max) are called “anim.kin”.

direct-drive

When direct-drive is present, the bogey animation is linked to the steam piston and physics system.

If this tag is not included the piston and steam sounds will not work!

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Track Sound.

TAGS:

kuid, trainz-build, category-class, category-region,

category-era, username, kind, direct-drive, alias, animdist, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: BRIDGE

Description

Road or rail bridges and similar assets, as variable length splines. The bridge kind may include initiator and terminator segments, and shadows. The height and gradient of the bridge spline may be varied in Surveyor.

Container Structure

A well formed bridge kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

bridge	
bendy	<i>data</i>
carrate	<i>data</i>
casts_shadows	<i>data</i>
endlength	<i>data</i>
grounded	<i>data</i>
isroad	<i>data</i>
istrack	<i>data</i>
length	<i>data</i>
repeats	<i>data</i>
rgb	<i>data</i>
shadows	<i>data</i>
upright	<i>data</i>
visible-on-minimap	<i>data</i>
width	<i>data</i>
kuid	<i>kuid</i>

trainz-build *data*

category-class *data*

category-region *data*

category-era *data*

username *data*

kind *data*

bridgetrack *kuid*

height *data*

trackoffsets *data*

initiator *data*

divider *data*

terminator *data*

kuid-table

0 *kuid*

thumbnails

0

image *file*

width *data*

height *data*

username

Asset username. This will be the name that appears in the surveyor menu.

height

Height from the track level to the base. Must be a negative value in order to raise the bridge above the ground.

kind

Must be "bridge".

rgb

This value should be left as default.

bridgetrack

Kuid of the track type to be used.

istrack\isroad

Two boolean tags detailing the behavior of the bridge. If the isroad is set to true, then cars are placed on the bridge. Both values should not be set to true.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

Additional Tags

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bend around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

grounded

Height in meters for the road to be offset from terrain.

length

Length of track segment in meters

repeats

The number of times the mesh is placed between spline

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be "TB - Bridge".

[See the "Track" section of the "Classes and Codes" appendix located at the end of this document.](#)

category-region

[See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.](#)

0 *kuid*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “BB - Buildable (Kind Buildable)”

See the “Buildings & Structures” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “buildable”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the buildable kind also contains additional containers and tags that are specialised to the requirements of the kind.

Attached Track Container

Auto-generated spline track. Generated through

attachment points located within the default mesh. Attached-tracks update automatically to the spline track connected to it. You may over-ride this auto-update feature by adding useadjoiningtracktype 0

Note. Correct track end attachment orientation is essential. The Y axis must point ‘out’ at the correct angle. The Z axis must point ‘up’.

The Attached Track Container has the following tags and containers:

track

Kuid of the track to be used.

useadjoiningtracktype

Indicates whether the track type should change to match that of the first track joined to the object.

Vertex

Attachment points at which to place track.

Attached Trigger Container

A Trigger is a point along an attached track with a specified radius. When a compatible rollingstock item enters this radius it triggers a set of commands, controlled through its script. A trigger is setup setup in an industries config.txt.

The Attached Trigger Container has the following tags:

att

The attachment point (stored in the mesh file) to place the trigger.

radius

Radius (in meters) of the trigger.

track

The track name which the train must be on to trigger.

Consists Container

The consists tag stores information on consists that can be generated by the industry.

The Consists Container has the following Tags:

show-in-consist-menu

Boolean flag that dictates whether this train appears in the consist menu (0 - false, 1 - true). The consist menu was along the bottom of the screen in the original Trainz and UTC but is no longer present. It effectively stopped a user from getting access to an AI train. Redundant for most uses except for legacy/scenario usage.

coupling-mask

Coupling mask that applies to the consist. 0 will block off all coupling activity while “1” will mean you can couple with a vehicle.

decoupling-mask

Coupling mask that applies to the consist. 0 will mean you can't decouple vehicles in the train while 1 means you can decouple vehicles.

Consist Element(Consist subcontainer)

vehicle

The kuid of the vehicle to be used.

facing

Indicates the direction of the vehicle.

running-number

Running number of the vehicle.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

Additional Tags

passenger-height

Used when making a station. Indicates the height of the platform on which the passengers stand.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Queues Container, Smoke Container, SoundScript Container, Mesh Table, Attached Track Container, Attached Trigger Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Consists Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autaname, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, surveyor-name-label, surveyor-only, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: CHUNKY-TRACK

Description

Track and rails for Trains (the common flexi-track) defining the cross section shape and properties of the track. Chunky-track uses a texture file but does not require a 3dsmax or gmax mesh model.

Container Structure

A well formed chunky-track kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

chunky-track

bendy	<i>data</i>
carrate	<i>data</i>
casts_shadows	<i>data</i>
endlength	<i>data</i>
grounded	<i>data</i>
isroad	<i>data</i>
istrack	<i>data</i>
length	<i>data</i>
repeats	<i>data</i>
rgb	<i>data</i>
shadows	<i>data</i>
upright	<i>data</i>
visible-on-minimap	<i>data</i>
width	<i>data</i>
kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

chunky_mesh *data*

chunky_info *data*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be set to "TR - Rails".

[See the "Classes and Codes" appendix located at the end of this document for more information.](#)

category-region

[See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be "track".

rgb

This value should be left as default.

Additional Tags

Kind bridge is derived from kind track and shares most of the same tags which are detailed in the [KIND TRACK](#) section of this chapter.

grounded

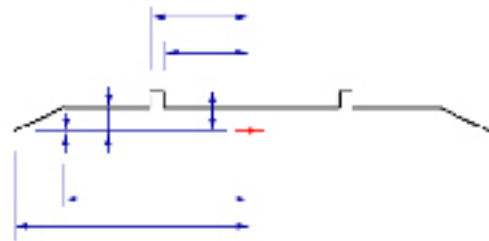
Height in meters for the road to be offset from terrain.

chunky_mesh

Name of texture to apply to rail. The texture must be within a directory of the same name (ie. "textureName\textureName.texture.txt"). The chunky_mesh value will simply be the name of this directory (ie. "textureName").w

chunky_info

These values (in metres) define the shape of the mesh created for the track. See drawing below:



chunky_info 0, 2, 1.2, 0.2, 0.85, 0.3, 0.7

chunky_info A, B, C, D, E, F, G

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bent around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

isroad

Specifies track is a road with cars, set to 1 for cars to appear on road.

istrack

0 = This is not rail tracks.

length

Length of track segment in meters

repeats

The number of times the mesh is placed between spline points

shadows

upright

Specifies whether the bridge “legs” point vertically, or perpendicular to the spline.

visible-on-minimap

Specifies whether the object\track is displayed on the minimap.

width

Width of track mesh in meters.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Track Sound, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

bendy, carrate, casts_shadows, endlength, grounded, isroad, istrack, length, repeats, rgb, shadows, upright, visible-on-minimap, width, kuid, trainz-build, category-class, category-region, category-era, username, kind, chunky_mesh, chunky_info, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, divider, dont-flip-terminator, hidden, initiator, invisible, license, light, organisation, terminator, uncached_alphas, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: DRIVERCHARACTER

Description

The locomotive driver character. This specifies the picture icon that appears in Driver as the engine driver.

Container Structure

A well formed drivercharacter kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

drivercharacter

kind	data
face-texture	file
kuid	kuid
mesh	kuid
username	data
trainz-build	data
category-class	data
category-region	data
category-era	data

kuid-table

0	kuid

thumbnails

0	
image	file
width	data
height	data
1	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “OHD - Locomotive Driver”.

[See the “Classes and Codes” appendix located at the end of this document for more information.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this](#)

[document for a list of valid category-era values.](#)

username

Asset username. This will be the text name of the driver.

kind

Must be “drivercharacter”.

Thumbnails

As well as the 240x180 image used as the assets preview thumbnail, a 32x32 thumbnail should be included as well. This is a half-size representation of the “face-texture” image, and is used where the small driver image is displayed. Not including this second thumbnail will cause Trainz to use the larger one.

face-texture

This is the driver icon used in TRS. Must be 64x64 pixels.

mesh

This refers to the kuid of the mesh asset inserted in to the locomotive mesh at a.driver0 (when in the Driver Module).

Kuid Table

The kuid of the driver mesh should be present in the kuid table.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, face-texture, mesh, alias, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

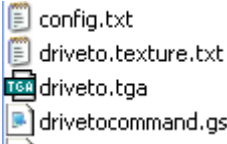
KIND: DRIVERCOMMAND

Description

A command for the train driver to accomplish a specific task.

Container Structure

A well formed drivercommand kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

drivercommand	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
supports-null-driver-character	data
class	data
script	data
thumbnails	
0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “YD - Driver Command”.

See the “Maps & Scenarios” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the driver menu.

kind

Must be “drivercommand”.

Class

The name of the scenario class within the script file.

Script

The script file (gs or gse file).

Additional Tags

supports-null-driver-character

Command can be executed without a driver present in the selected loco.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, supports-null-driver-character, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

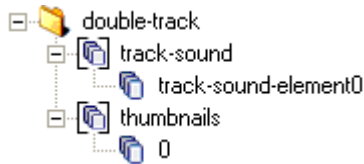
KIND: DOUBLE-TRACK

Description

Track splines that may place two or more tracks as one model, by specifying the track spacings to be used.

Container Structure

A well formed double-track kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

double-track	
bendy	data
carrate	data
casts_shadows	data
endlength	data
grounded	data
isroad	data
istrack	data
length	data
repeats	data
rgb	data
shadows	data
upright	data
visible-on-minimap	data
width	data
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
bridgetrack	kuid

height *data*

trackoffsets *data*

kuid-table

0 *kuid*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be "TR - Rails".

[See the "Track" section of the "Classes and Codes" appendix located at the end of this document.](#)

category-region

[See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be "bridge".

rgb

This value should be left as default.

trackoffsets

Distance in meters the rail/s are attached to the center of the spline.

Additional Tags

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bend around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

grounded

Height in meters for the road to be offset from terrain.

isroad

Specifies track is a road with cars, set to 1 for cars to appear on road.

istrack

0 = This is not rail tracks.

length

Length of track segment in meters

repeats

The number of times the mesh is placed between spline points

shadows

BLAH

upright

Specifies whether the bridge "legs" point vertically, or perpendicular to the spline.

visible-on-minimap

Specifies whether the object\track is displayed on the minimap.

width

Width of track mesh in meters.

bridgetrack

Kuid for the type of rail or road used on bridge.

height

Height from the track level to the base. Should be negative.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.



THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: ENGINE

Description

An engine specification for locomotives and rolling-stock which defines the detailed performance requirements; including throttle requirements and engine and braking performance.

Container Structure

A well formed engine kind has the following container structure:

See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

engine	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
flowsize	
trainbrakepipe	data
epreservoirpipe	data
no3pipe	data
no4pipe	data
auxreservoirvent	data
auxreservoir_no3	data

auxreservoir_trainbrakepipe	<i>data</i>		independantbrakecylinder	
autobrakecylindervent	<i>data</i>	<i>data</i>		
auxreservoir_autobrakecylinder	<i>data</i>		pressure	
equaliser_mainreservoir	<i>data</i>		scale	<i>data</i>
equaliservent	<i>data</i>		compressor	<i>data</i>
equaliserventhandleoff	<i>data</i>		mainreservoir	<i>data</i>
equaliserventemergency	<i>data</i>		highspeedexhauster	<i>data</i>
no3pipeevent	<i>data</i>		brakepipe	<i>data</i>
no3pipe_mainreservoir	<i>data</i>		brakeinitial	<i>data</i>
compressor	<i>data</i>		brakefull	<i>data</i>
trainbrakepipe_reservoir	<i>data</i>		indbrakefull	<i>data</i>
trainbrakepipeevent	<i>data</i>		trainbrakepipe_start	<i>data</i>
no3pipe_autobrakecylinder	<i>data</i>		epreservoirpipe_start	<i>data</i>
epreservoirpipe_autobrakecylinder	<i>data</i>		no3pipe_start	<i>data</i>
mainreservoir_ep	<i>data</i>		no4pipe_start	<i>data</i>
vacuumbrakepipe	<i>data</i>		auxreservoir_start	<i>data</i>
vacuumbrakepipereleaseevent	<i>data</i>		autobrakecylinder_start	<i>data</i>
vacuumbrakepipeevent	<i>data</i>		vacuumbrakepipe_start	<i>data</i>
vacuumbrakereservoir_vacuumbrakepipe	<i>data</i>		vacuumbrakereservoir_start	<i>data</i>
vacuumbrakecylinder_vacuumbrakepipe	<i>data</i>		vacuumbrakecylinder_start	<i>data</i>
highspeedexhauster_vacuumbrakepipe	<i>data</i>		mainreservoir_start	<i>data</i>
volume			equaliser_start	<i>data</i>
scale	<i>data</i>		independantbrakecylinder_start	<i>data</i>
trainbrakepipe	<i>data</i>		mass	
epreservoirpipe	<i>data</i>		scale	<i>data</i>
no3pipe	<i>data</i>		fuel	<i>data</i>
no4pipe	<i>data</i>		motor	
auxreservoir	<i>data</i>		resistance	<i>data</i>
autobrakecylinder	<i>data</i>	<i>data</i>	adhesion	
vacuumbrakepipe			maxvoltage	<i>data</i>
vacuumbrakereservoir	<i>data</i>		maxspeed	<i>data</i>
vacuumbrakecylinder	<i>data</i>		brakeratio	<i>data</i>
mainreservoir	<i>data</i>		max-accel	<i>data</i>
equaliser	<i>data</i>		max-decel	<i>data</i>
			throttle-notches	<i>data</i>

axle-count	<i>data</i>
surface-area	<i>data</i>
moving-friction-coefficient	<i>data</i>
air-drag-coefficient	<i>data</i>

throttle-power

***See Chapter 7 Examples for various Throttle-Power values.**

dynamic-brake

***See Chapter 7 Examples for various Dynamic Brake values.**

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following:

“AA - Electric Multi-current”, “AC - AC Electric”, “AD - DC Electric”, “AE - Experimental or Special”, “AG - Gas Turbine”, “AH - Diesel Hydraulic”, “AL - Diesel & Diesel Electric”, “AM - Mammal”, “AS - Steam Loco & Tender”, “AT - Steam Tank”.

See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn't appear in

the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “engine”.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the engine kind also contains additional containers and tags that are specialised to the requirements of the kind.

Throttle-Power Container

Contains acceleration variables as used in cabin mode.

1

0	30
5	25 = At speed 5, acceleration = 25
10	15
12	0

See Chapter 7 for example Throttle-Power values.

Dynamic Brake Container

Contains deceleration variables for dynamic braking in cabin mode.

1

1.333	0
2	30
5	25 = At speed 5, deceleration = 25
10	15
12	0

See Chapter 7 for example Dynaic Brake values.

Pressure Container

Stores brake system pressures.

The Pressure Container has the following tags:

scale

Multiplies pressure by given value, generally leave this setting.

compressor

(120psi expressed in grams/m³) Compressor maximum pressure.

mainreservoir

Main reservoir maximum pressure

highspeedexhauster

For vacuum braking - not currently in use, generally leave this setting.

brakepipe

(80psi expressed in grams/m³) Brake pipe pressure when fully charged

brakeinitial

(72psi expressed in grams/m³) Brake pipe pressure after initial service reduction (for self lapping brakes)

brakefull

(57psi expressed in grams/m³) Brake pipe pressure after full service reduction (for self lapping brakes)

indbrakefull

Brake cylinder pressure for independant brake service

trainbrakepipe_start

Brake pipe pressure on loading the game.

epreservoirpipe_start

For electro pneumatic braking - not currently in use, generally leave this setting.

no3pipe_start

Generally leave these settings.

no4pipe_start

Generally leave these settings.

auxreservoir_start

Auxiliary reservoir pressure on loading the game

autobrakecylinder_start

Train brake cylinder pressure on loading the game

vacuumbrakepipe_start

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakereservoir_start

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakecylinder_start

For vacuum braking - not currently in use, generally leave this setting.

mainreservoir_start

(100psi expressed in grams/m³) Main Reservoir pressure on loading the game.

equaliser_start

Equalising Reservoir pressure on loading the game.

independantbrakecylinder_start

Locomotive brake cylinder pressure on loading the game.

Mass Container

The mass container stores information related to fuel consumption. These tags aren't in use and shouldn't generally be used.

The mass container has the following tags:

scale

Multiplies fuel mass by given value, not currently in use, generally leave this setting.

fuel

Fuel level, not currently in use, generally leave this setting.

Motor Container

The Motor Container stores an assortment of values related to motor function, particularly that of DCC.

resistance

Power figure for DCC, higher resistance value=less power

adhesion

Adhesion parameter, higher value=greater adhesion

maxvoltage

Generally leave this setting

maxspeed

Maximum speed for DCC, expressed in metres per second.

brakeratio

Brake force for pressure reduction.

max-accel

Parameters for DCC acceleration & deceleration.

max-decel

Parameters for DCC acceleration & deceleration.

throttle-notches

Number of throttle notches.

axle-count

Resistance - Axle Count.

surface-area

Resistance - Surface Area.

moving-friction-coefficient

Resistance - Moving friction.

air-drag-coefficient

Resistance - Air drag.

Flowsize Container

Flowsize settings specify the rate of flow through the pipes. Generally these setting should be left unaltered.

The Flowsize Container has the following tags:

trainbrakepipe

Flowsize of the brake pipe.

epreservoirpipe

Flowsize of the electric pneumatic braking

no3pipe

Flowsize of the independent brake pipe.

no4pipe

Flowsize of the bail pipe.

auxreservoirvent

Flowsize of the auxiliary reservoir vent.

auxreservoir_no3

Flowsize of the auxiliary independent brake pipe.

auxreservoir_trainbrakepipe

Flowsize of the auxiliary reservoir brake pipe.

autobrakecylindervent

Flowsize of the automatic brake cylinder vent.

auxreservoir_autobrakecylinder

Flowsize of the auxiliary reservoir automatic brake cylinder.

equaliser_mainreservoir

Flowsize of the equaliser main reservoir.

equaliservent

Flowsize of the equaliser vent.

equaliserventhandleoff

Flowsize of the equaliser to the atmosphere when in the "handle off" position.

equaliserventemergency

Flowsize of the emergency equaliser vent.

no3pipevent

Flowsize of the independent brake pipe.

no3pipe_mainreservoir

Flowsize of the independent brake main reservoir.

compressor

Flowsize of the compressor.

trainbrakepipe_reservoir

Flowsize of the brake pipe reservoir.

trainbrakepipevent

Flowsize of the brake pipe vent.

no3pipe_autobrakecylinder

Flowsize of the independent automatic brake pipe cylinder.

epreservoirpipe-autobrakecylinder

Flowsize of the electro pneumatic automatic brake cylinder reservoir.

mainreservoir_ep

Flowsize of the electro pneumatic main reservoir.

vacuumbrakepipe

Flowsize of the vacuum brake pipe.

vacuumbrakepipereleasevent

Flowsize of the vacuum brake pipe release vent.

vacuumbrakepipevent

Flowsize of the vacuum brake pipe vent.

vacuumbrakereservoir_vacuumbrakepipe

Flowsize of the vacuum brake pipe reservoir.

vacuumbrakecylinder_vacuumbrakepipe

Flowsize of the vacuum brake pipe cylinder.

highspeedexhauster_vacuumbrakepipe

Flowsize of the high speed exhauser vacuum brake pipe.

Volume Container

The volume container stores information regarding the size of pipes and appliances. Generally these settings should remain unaltered.

The Volume Container has the following tags:

scale

multiplies volume by given value, generally leave this setting.

trainbrakepipe

brake pipe volume

epreservoirpipe

For electro pneumatic braking - not currently in use, generally leave this setting.

no3pipe

Independent brake pipe

no4pipe

Bail pipe - not currently in use, generally leave this setting.

auxreservoir

Auxiliary reservoir volume.

autobrakecylinder

Brake cylinder volume.

vacuumbrakepipe

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakereservoir

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakecylinder

For vacuum braking - not currently in use, generally leave this setting.

mainreservoir

Main reservoir volume.

equaliser

Equalising reservoir volume

independantbrakecylinder

Loco brake cylinder volume

Notes

Equalisation of Pressures

There is a point at which no further brake pipe pressure reduction will result in increased braking effort, this is known as full application or equalisation of pressures.

Imagine you made a 26 psi reduction when operating a loco with a 90psi brake pipe. 90psi in the train pipe minus 26psi reduction equals 64 psi in the pipe. Due to the 2.5:

1 ratio of auxiliary reservoir volume to brake cylinder volume, the 26 psi reduction puts 64 psi into the brake cylinder.

As the pressure in the reservoir and the pressure in the cylinder is now equal, no more air will flow into the brake cylinder; and making a further reduction in brake pipe pressure will have no effect on braking.

Equalisation occurs at different pressures, depending on the train pipe feed pressure.

100 psi pipe (e.g. the UK locos - 7 bar) equalisation at 71 psi.

90 psi pipe (e.g. the US locos) equalisation at 64 psi. 72 psi pipe (e.g. French & Queensland locos) equalisation at 49 psi.

The easiest way to set your custom content to the desired brake pipe feed pressure is to copy the entire pressure section from the config of a loco that uses the pressure you desire.

*Note: Converting PSI to Grams /m cubed...

e.g. 90psi... $(90+14.7).0000703$

$104.7 \times .0000703=.00736041$

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Throttle Power Container, Dynamic Brake Container, Pressure Container, Mass Container, Motor Container, Flowsize Container, Volume Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: ENGINESOUND

Description

An engine sound specification, detailing the locomotive engine sound files referenced by the enginesound tag in a traincar kind.

Container Structure

A well formed enginesound kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

enginesound

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ZS - Enginesound”.

See the “Train Parts” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “enginesound”.

Notes

Though not mentioned in the config.txt itself, enginesounds must be named in a specific way in order to work correctly.

Diesel and Electric

“down 2 - 1.wav”, “down 3 - 2.wav”, “down 4 - 3.wav”, “down 5 - 4.wav”, “down 6 - 5.wav”, “down 7 - 6.wav”, “down 8 - 7.wav”

“idle 1.wav”, “idle 2.wav”, “idle 3.wav”, “idle 4.wav”, “idle 5.wav”, “idle 6.wav”, “idle 7.wav”, “idle 8.wav”

“stop.wav”

“up 1 - 2.wav”, “up 2 - 3.wav”, “up 3 - 4.wav”, “up 4 - 5.wav”, “up 5 - 6.wav”, “up 6 - 7.wav”, “up 7 - 8.wav”

Steam

These file are the steam engine idling sounds played after the steam engine is stationary for 1, 2 and 3 minutes.

“loco-stationary.fast.wav” (1 minute)

“loco-stationary.med.wav” (2 minutes)

“loco-stationary.slow.wav” (3 minutes)

Piston stroke sounds, played every 180 degrees revolution of the piston wheel played in sequence and repeated up to about 40 kph.

“piston_stroke1.wav”, “piston_stroke2.wav”, “piston_stroke3.wav”, “piston_stroke4.wav”

From 40 kph upwards, the following sound loop is cross-faded as the piston sounds die off. The loop is pitched shifted (through code) relative to the locomotive’s velocity.

“steam_loop.wav”

The general hiss from the smoke stack:

“smoke_stack_hiss.wav”

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: ENVIRONMENT

Description

Additional sky textures, specifying the normal, night and stormy sky images to be used in Trainz.

Container Structure

A well formed environment kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

environment

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
normal	<i>data</i>

storm *data*

night *data*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ES - Sky”.

[See the “Environment” section of the “Classes and Codes” appendix located at the end of this document.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “environment”.

Additional Tags

normal

Name of image file for normal sky. File should be 256 x 256 pixel 24bit tga.

The file extension should be excluded here, ie “**QLD_Sky**” rather than “QLD_Sky.texture.txt”.

storm

Name of image file for stormy sky. File should be 256 x 256 pixel 24bit tga.

The file extension should be excluded here, ie “QLD_Sky-Storm” rather than “QLD_Sky.texture.txt”.

night

Name of image file for night sky. File should be 256 x 256 pixel 24bit tga.

The file extension should be excluded here, ie “QLD_Sky-Night” rather than “QLD_Sky.texture.txt”.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, normal, storm, night, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: FIXEDTRACK

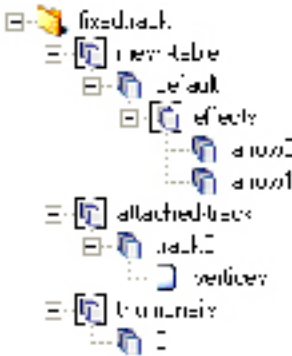
Description

A fixedtrack asset can be likened to a model trains sectional track system. The models may be straight or curved and snap into position when moved onto another track in Surveyor.



Container Structure

A well formed fixedtrack kind has the following container structure:



See Chapter 7 for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

fixedtrack	
username	data
kind	data
kuid	kuid
preview-mesh-kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
mesh-table	
default	
mesh	file
auto-create	data
effects	
arrow0	
att	data
default-mesh	kuid
surveyor-only	data
kind	data
arrow1	
att	data
default-mesh	kuid
surveyor-only	data
kind	data
attached-track	

track0

track *kuid*

vertices

0 *data*

1 *data*

2 *data*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “TF - Fixed Track”.

See the “Classes and Codes” appendix located at the end of this document for more information.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “fixedtrack”.

preview-mesh-kuid

the mesh to be used in the surveyor preview area. This is useful when an asset has a large bounding box. Ie the “Airport” with it's jet animation.

Mesh Table

Default

Contains the default mesh.

arrow0, arrow1

These attachment effects place an arrow mesh at each end of the Default Mesh according to the attachment points stored in the mesh file (and referenced in the vertices container). These arrows are used as guides and only shown in surveyor.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the fixedtrack kind also contains additional containers and tags that are specialised to the requirements of the kind.

Attached Track Container

Auto-generated spline track. Generated through attachment points located within the default mesh. Attached-tracks update automatically to the spline track connected to it. You may over-ride this auto-update feature by adding useadjoiningtracktype 0

Note. Correct track end attachment orientation is essential. The Y axis must point ‘out’ at the correct angle. The Z axis must point ‘up’.

The Attached Track Container has the following tags and containers:

track

Kuid of the track to be used.

useadjoiningtracktype

Indicates whether the track type should change to match that of the first track joined to the object.

Vertex

Attachment points at which to place track.

Junction-Vertices Container

The Junction-Vertices Container contains the tags needed to handle the lever portions of a fixed track.

The Junction-Vertices Container has the following tags:

junction-lever-mesh

The mesh (selected from the mesh table) to be used as a junction lever.

junction-vertex

The attachment point (located in mesh file) at which to place the lever.

Additional Tags

preview-mesh-kuid

Each fixedtrack asset needs a preview-mesh a spline tracks will not render in the Preview window.

A preview-mesh can simply be setup as a kind mesh. This way the preview-mesh will never be selectable or seen in Surveyor.

use-gradient-track

This means to use the spline gradient rather than following the ground height.

Notes

Technically, all a fixedtrack comprises of is a mesh asset with an attached track (or tracks) and surveyor only rendered arrows so the user knows where the fixedtrack starts and ends.

The model comprises simply of a few attachment points (using the a.name naming convention) set-up accurately in Max or Gmax, and a single invisible polygon to allow exporting, and for in-game asset selection.

Note. Correct track end attachment orientation is essential. The Y axis must point 'out' at the correct angle. The Z axis must point 'up'. Mid points just need to be in the correct spline path. See diagram below. TRS released fixedtracks comprise of only curved and straight sections. Crossings and junctions are easily created using the attached-track set-up. For crossings, just create two attached-track fields. For junctions do the same but use one of the attachments twice as shown below (where a.track0a is used twice).

For example:

attached-track

track0

track	<KUID:-1:15>
useadjoiningtracktype	0

vertices

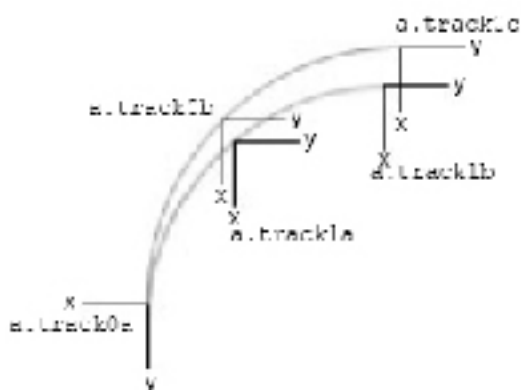
0	a.track0a
1	a.track0b
2	a.track0c

track1

track	<KUID:-1:15>

vertices

0	a.track0a
1	a.track1a
2	a.track1b



THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, SoundScript Container, Attached Track Container, Attached Trigger Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Junction-Vertices Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autoname, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, surveyor-name-label, surveyor-only, user-gradient-track, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: GROUNDTEXTURE

Description

A ground texture is tiled in Surveyor to color and cover the base grid. It can optionally reference a low polygon mesh and insert the mesh automatically as the ground is painted.

Container Structure

A well formed groundtexture kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

groundtexture

kuid	<i>kuid</i>
kind	<i>data</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
username	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
texture	<i>file</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “G - Ground”, “GA - Arid”, “GL - Lush”, “GS - Seasonal”.

See the “Ground” section of the “Classes and Codes” appendix located at the end of this document for more information.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears as a mouseover hint in the surveyor menu. Textures are also sorted alphabetically by username.

kind

Must be “groundtexture”.

texture

The texture file. Should be a 128x128pixel bitmap.

Additional Tags

clutter-mesh

Ground textures can now reference a mesh and insert the mesh automatically as the ground is painted.

Painting over a clutter-mesh ground texture effectively deletes clutter meshes and texture. The mesh it refers to is can be standard scenery object kind mesh.

Clutter-meshes must have only one Max material assigned to it only.

Polycounts must be very low.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, texture, author, category-keyword, clutter-mesh, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: HORNSOUND

Description

A traincar horn sound, referenced by the hornsound tag in a traincar config file. It references the various sound files to be used.

Container Structure

A well formed hornsound kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

hornsound

kind	<i>data</i>
kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
three-part	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in [Chapter 5](#):

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ZH - Hornsound”.

See the “Train Parts” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “hornsound”.

Additional Tags

two-part

Indicated that the Railyard and Driver hornsounds are different. The Driver hornsound is looping. If this tag is not present, the hornsound defaults to UTC equivalent non-looping format.

See [Chapter 7](#) for an example of a two-part hornsound.

three-part

Specifies that the hornsound has a beginning, middle and ending sound.

See [Chapter 7](#) for an example of a three-part hornsound.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, three-part, two-part, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

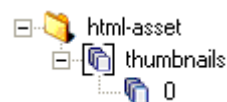
KIND: HTML-ASSET

Description

An html-asset example is the ingame tutorial. The config.txt file references one or more .html pages. The html-asset can be referenced from the scripts and from some of the Surveyor rules.

Container Structure

A well formed html-asset kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

html-asset

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be "YH - HTML-Asset"

See the "Maps & Scenarios" section of the "Classes and Codes" appendix located at the end of this document.

category-region

See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.

category-era

See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn't appear in the Surveyor menu, the username is used to identify the asset in the "Content Manager Plus" and "Content Creator Plus" programs.

kind

Must be "html-asset".

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

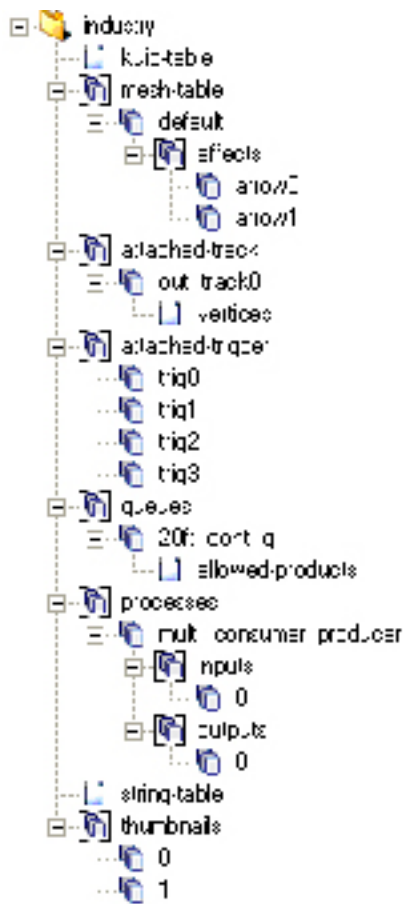
KIND: INDUSTRY

Description

A scenery asset with product processing functionality. Industry assets interact with compatible rolling stock assets through their script file and asset triggers. An Industry asset supports product queues and attached track.

Container Structure

A well formed industry kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

industry			
kuid		kuid	
trainz-build		data	
category-class		data	
category-region		data	
category-era		data	
username		data	
kind		data	
script		file	
			class
			data
			preview-mesh-kuid
			kuid
			icon0
			kuid
			kuid-table
			coal
			kuid
			mesh-table
			default
			mesh
			file
			auto-create
			data
			attached-track
			out-track0
			track
			kuid
			vertices
			0
			data
			attached-trigger
			trig0
			att
			data
			radius
			data
			queues
			20ft-cont-q
			size
			data
			initial-count
			data
			product-kuid
			kuid
			allowed-products
			0
			kuid
			processes
			multi-consumer-producer
			start-enabled
			data
			duration
			data
			inputs
			0
			amount
			data
			queue
			data
			outputs
			0

amount *data*

queue *data*

string-table

multi_pickupdropoff *data*

thumbnails

0

image *file*

width *data*

height *data*

1

image *file*

width *data*

height *data*

kind

Must be "industry".

class

The name of the scenario class within the script file.

script

The script file (gs or gse file).

icon0

Kuid of the preview icon. Should be a 32x32 TGA.

preview-mesh-kuid

the mesh to be used in the surveyor preview area. This is useful when an asset has a large bounding box. Ie the "Airport" with it's jet animation.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

String Table

The string table stores a list of text strings to be used by the industry script.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the industry kind also contains additional containers and tags that are specialised to the requirements of the kind.

Attached Track Container

Auto-generated spline track. Generated through attachment points located within the default mesh. Attached-tracks update automatically to the spline track connected to it. You may over-ride this auto-update feature by adding useadjoiningtracktype 0

Note. Correct track end attachment orientation is essential. The Y axis must point 'out' at the correct angle. The Z axis must point 'up'.

The Attached Track Container has the following tags and containers:

track

Kuid of the track to be used.

useadjoiningtracktype

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following:

"BIN - Industry asset with product processing functionality", "BPF - Passenger Station with passenger processing functionality", "BPN Passenger Station (non-functional)", "BB Buildable (Kind Buildable)"

See the "Buildings & Structures" section of the "Classes and Codes" appendix located at the end of this document.

category-region

See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.

category-era

See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

Indicates whether the track type should change to match that of the first track joined to the object.

Vertex

Attachment points at which to place track.

Attached Trigger Container

A Trigger is a point along an attached track with a specified radius. When a compatible rollingstock item enters this radius it triggers a set of commands, controlled through its script. A trigger is setup in an industries config.txt.

The Attached Trigger Container has the following tags:

att

The attachment point (stored in the mesh file) to place the trigger.

radius

Radius (in meters) of the trigger.

track

The track name which the train must be on to trigger.

Consists Container

The consists tag stores information on consists that can be generated by the industry.

The Consists Container has the following Tags:

show-in-consist-menu

Boolean flag that dictates whether this train appears in the consist menu (0 - false, 1 - true). The consist menu was along the bottom of the screen in the original Trainz and UTC but is no longer present. It effectively stopped a user from getting access to an AI train. Redundant for most uses except for legacy/scenario usage.

coupling-mask

Coupling mask that applies to the consist. 0 will block off all coupling activity while "1" will mean you can couple with a vehicle.

decoupling-mask

Coupling mask that applies to the consist. 0 will mean you can't decouple vehicles in the train while 1 means you can decouple vehicles.

Consist Element(Consist subcontainer)

vehicle

The kuid of the vehicle to be used.

facing

Indicates the direction of the vehicle.

running-number

Running number of the vehicle.

Processes Container

Processes (required) The input and output settings of the industry. You can specify the amount of input and output for each queue referenced product as well as the duration (or rate) in seconds for that process to take place.

All queues and processes are linked through the industry asset's script file.

The Processes Container has the following tags and containers:

start-enabled

Specifies whether the process starts enabled.

duration

Length of time (in seconds) that the process runs for.

Inputs Container (Processes subcontainer)

amount

Amount required as input.

queue

Queue from which to take input.

Outputs Container(Processes subcontainer)

amount

Amount to output.

queue

Queue in which to place output.

Queue Container

size: Size of queue.

animated-mesh: Animated mesh which changes as the queue becomes full.

custom-attachments: Not used.

initial-count: The initial number of items in the queue.

product-kuid: The product type used to fill 'initial-count'

allowed products: The allowed products in this queue.

conflicts-with-queues: This queue and the conflicting queue(s) cannot be used simultaneously.

attachment-points: List of attachment points for this queue on which products are visualised. (Use this, OR animated-mesh)

allowed-categories: The allowed product categories in

this queue.

Notes

Perhaps the most simple examples of industry functionality are the TRS released Coalmine and the Powerstation assets.

When the coal hopper enters the trigger radius of the coalmine loading bay, it's script interacts with the hoppers own script. Particle effects (pfx) from the coalmine visually display the coal entering the hopper and the hopper animated load rises to show it's full state. The coalmine's own animated load pile reduces as does it's commodity level.

Similarly, when the full hopper enters the Powerstation trigger radius, the hopper's animated load lowers, the side doors open and the pfx on the hopper itself initiate. The animated load on the Powerstation increases and it's commodity level increases.

The hopper pfx, and the animated doors are both controlled by the hopper.gs script file.

[Sounds events and generic events can be linked to an animation key-frame to give great control over sound and script timing for industry and scenery assets.](#)

* Note: The increasing use of scripts in TRS adds huge flexibility and control to assets and their functionality.

Adding scripts on a per asset basis is a logical progression in the development of Trainz. However, we do

understand that most 3D modelers do not know a great deal about scripts let alone how to write them. With the

release of TRS, we are really dawning on a new era in Trainz custom content creation.

Because of this, we recognise there will need to be far more collaboration and group efforts for custom industry

asset creation. There are several very good script writers in the Trainz content Creation community. Just ask around the forums.

category-era, username, kind, alias, author, autoname, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-xyz-range, rotstep, script, snapgrid, snapmode, surveyor-name-label, surveyor-only, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Queues Container, Smoke Container, SoundScript Container, Mesh Table, Attached Track Container, Attached Trigger Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Processes Container, Consists Container.

TAGS:

kuid, trainz-build, category-class, category-region,

KIND: INTERIOR

Description

A traincar interior asset. It allows the interior mesh model to be defined, and has attached levers and controls to operate a locomotive in cab model. It also creates an interior for rolling stock.

Container Structure

A well formed interior kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

interior	
kuid	kuid
trainz-build	data

category-class	data
category-region	data
category-era	data
username	data
kind	data
cameradefault	data
cameralist	
camera0	data
camera1	data
camera2	data
camera3	data
camera4	data
camera5	data
mesh-table	
ampmeter_needle	
kind	data
mesh	file
att	data
limits	data
angles	data
att-parent	data
flow_needle	
kind	data
mesh	file
att	data
limits	data
att-parent	data
bploco_equaliser	
kind	data
mesh	file
att	data
limits	data
att-parent	data
bplocomain_needle	
kind	data

mesh	<i>file</i>	limits	<i>data</i>
att	<i>data</i>	angles	<i>data</i>
limits	<i>data</i>	notches	<i>data</i>
att-parent	<i>data</i>	notchheight	<i>data</i>
bptrainbrakecylinder_needle		mousespeed	<i>data</i>
kind	<i>data</i>	att-parent	<i>data</i>
mesh	<i>file</i>	reverser_lever	
att	<i>data</i>	kind	<i>data</i>
limits	<i>data</i>	mesh	<i>file</i>
att-parent	<i>data</i>	att	<i>data</i>
bptrainbrakepipe_needle		limits	<i>data</i>
kind	<i>data</i>	angles	<i>data</i>
mesh	<i>file</i>	notches	<i>data</i>
att	<i>data</i>	notchheight	<i>data</i>
limits	<i>data</i>	att-parent	<i>data</i>
att-parent	<i>data</i>	throttle_lever	
speedo_needle		kind	<i>data</i>
kind	<i>data</i>	mesh	<i>file</i>
mesh	<i>file</i>	att	<i>data</i>
att	<i>data</i>	limits	<i>data</i>
limits	<i>data</i>	angles	<i>data</i>
att-parent	<i>data</i>	notches	<i>data</i>
dynamicbrake_lever		notchheight	<i>data</i>
kind	<i>data</i>	mousespeed	<i>data</i>
mesh	<i>file</i>	att-parent	<i>data</i>
att	<i>data</i>	trainbrakelap_lever	
limits	<i>data</i>	kind	<i>data</i>
angles	<i>data</i>	mesh	<i>file</i>
notches	<i>data</i>	att	<i>data</i>
notchheight	<i>data</i>	limits	<i>data</i>
att-parent	<i>data</i>	angles	<i>data</i>
independantbrake_lever		notches	<i>data</i>
kind	<i>data</i>	notchheight	<i>data</i>
mesh	<i>file</i>	mousespeed	<i>data</i>
att	<i>data</i>	att-parent	<i>data</i>

horn

kind	<i>data</i>
mesh	<i>file</i>
att	<i>data</i>
limits	<i>data</i>
angles	<i>data</i>
notches	<i>data</i>
notchheight	<i>data</i>
mousespeed	<i>data</i>
att-parent	<i>data</i>

light_switch

kind	<i>data</i>
mesh	<i>file</i>
att	<i>data</i>
limits	<i>data</i>
angles	<i>data</i>
notches	<i>data</i>
notchheight	<i>data</i>
mousespeed	<i>data</i>
radius	<i>data</i>
att-parent	<i>data</i>

ampmeter

mesh	<i>file</i>

bploco

mesh	<i>file</i>

bptrain

mesh	<i>file</i>

brakepressure

mesh	<i>file</i>

chair

mesh	<i>file</i>

controlstand

mesh	<i>file</i>

horizblinds

mesh	<i>file</i>

interior_main

mesh	<i>file</i>

speedo

mesh	<i>file</i>

westinghouse

mesh	<i>file</i>

windows

mesh	<i>file</i>
opacity	<i>data</i>

wheelslip_light

kind	<i>data</i>
mesh	<i>file</i>
att	<i>data</i>
att-parent	<i>data</i>

default

mesh	<i>file</i>
auto-create	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ZI - Interior”.

See the “Train Parts” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

kind

Must be “interior”.

Mesh Table

kind

The type of interior object the particular mesh is. Affects the behavior of the mesh in game.

Kinds

lever

Levers, switches, dials etc

animated-lever

Animated Levers etc Eg. in steam cabs

collision-proxy

Mouse collisions for animated levers

needle

Gauge needles (I.e. Speedo, brake pres.)

pullrope

Pull rope horn as in the F7

light

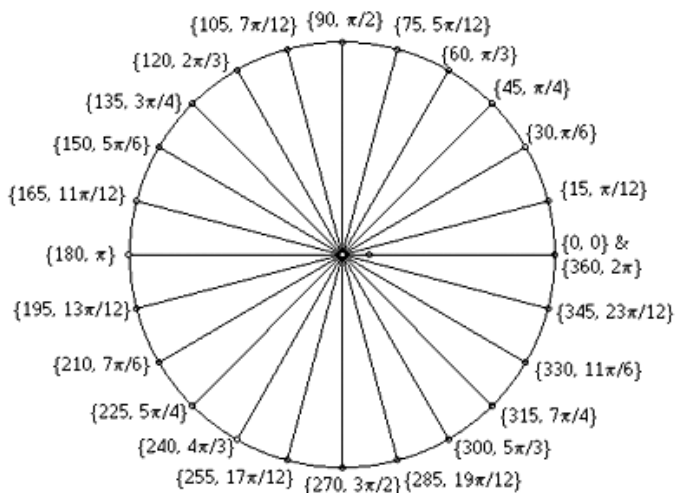
Wheelslip light

limits

Mathematical boundaries Trainz uses to determine the objects function. These values vary as different objects use different mathematical units.

angles

Rotational boundaries in radians relative to its attachment point. Refer to the radian/degree circle diagram below:



notches

The position of notches within the angle boundaries. These are represented as decimal points between and including 0 and 1.

notchheight

The size of the notches specified.

radius

The notch position relative to the attachment point.

mousespeed

This controls the use of the mouse on screen. Use this to control the mouse speed and push/pull direction for levers and dials.

- mousespeed -1 Inverts mouse direction.
- mousespeed 2 Doubles mouse speed in default direction.
- mousespeed -0.5 Inverts mouse direction and halves the speed.

test-collisions

Mouse cannot be used for this mesh. Collision mesh used instead. I.e. animated-levers.

opacity

Usually used for the window mesh to give transparency (and the impression of reflection).

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the interior kind also contains additional containers and tags that are specialised to the requirements of the kind.

CameraList Container

Contains a list of camera coordinates for the interior cameras relative to a.cabfront.

camera

A camera contains 5 numeric coordinates that determine the placement and orientation of the camera. These are:

0,0,0,0,0 =left/right, front/back, up/down, yaw, pitch

To determine these variables add -freeintcam to the trainzoptions.txt. Pan around the interior using arrow keys and mouse.

Co-ordinates are displayed at bottom-left of screen.

Additional Tags

cameradefault



The in-cab camera view Trainz defaults to when entering the cab.

Notes

[Extra notes on modelling interiors can be found in Chapter 8.](#)

Interior Attachment types

These values should be used as the name of your mesh table entries when constructing an interior (ie a throttle should be called "throttle_lever").

pantograph_lever

Pantograph lever/switch. For raising and lowering pantographs on electric locos.

horn

Locomotive's horn.

independantbrake_lever

Independent (Loco) brake lever

reverser_lever

Reverser lever (Forward/Neutral/Reverse)

throttle_lever

Throttle / power handle

trainbrake_lever

Train brake lever - self lapping

trainbrakelap_lever

Train brake lever with lap position.

dynamicbrake_lever

For selecting dynamic brake

bplocomain_needle

Main reservoir pressure needle

bploco_equalizer

Equalising reservoir pressure needle

bptrainbrakepipe_needle

Brake pipe pressure needle

bptrainbrakecylinder_needle

Brake cylinder pressure needle

speedo_needle

Speedometer needle

ampmeter_needle

Power meter needle

flow_needle

Flow gauge needle

windows

Textured mesh with low opacity (semi-transparent) to give impression of reflection. This mesh has the same 3D origin point as the main .im mesh, therefore does not require an attachment point.

wheelslip_light

A warning light mesh that is only visible when the locomotive loses traction. This mesh has the same 3D origin point as the main .pm model, therefore does not require an attachment point

switch0, switch1 etc

Switches

light_switch

Headlight switch

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, CameraList, SoundScript Container, Mesh Table.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, cameradefault, alias, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: LIBRARY

Description

Coded modules that interact with other coded modules.

Container Structure

A well formed library kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

library

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
class	<i>data</i>
script	<i>file</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “YR - Rule”.

See the “Maps & Scenarios” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “library”.

Script

The script file (gs or gse file).

Class

The name of the scenario class within the script file.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: MESH

Description

A mesh that is never referenced through Surveyor panels but is referenced from another asset. It could be referenced through the preview-mesh-kuid tag or as a kind attachment effect, like the red arrows used on fixed track assets.

Container Structure

A well formed mesh kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mesh

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

mesh-table

default

mesh	<i>file</i>
auto-create	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in [Chapter 5](#):

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “HM - Mesh”.

[See the “Mesh” section of the “Classes and Codes” appendix located at the end of this document.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “mesh”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, script, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

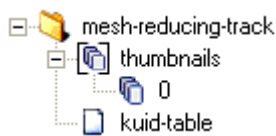
KIND: MESH-REDUCING-TRACK

Description

Mesh-Reducing-Track is used to create poly efficient splines. The asset consists of a short high detailed mesh and a longer less detailed mesh based on the same object. The short mesh is displayed when the camera is close to the asset whilst the long mesh is shown when less detail is required.

Container Structure

A well formed mesh-reducing-track kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mesh-reducing-track

carrate	data
casts_shadows	data
endlength	data
grounded	data
isroad	data
istrack	data
length	data
repeats	data
rgb	data
shadows	data
upright	data
visible-on-minimap	data
width	data
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data

unit_mesh	data
bendy	data
thumbnails	
0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be set to “TR - Rails”.

[See the “Classes and Codes” appendix located at the end of this document for more information.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “Track”.

rgb

This value should be left as default.

grounded

Height in meters for the road to be offset from terrain. Eg 0.4..

length

Length of track segment in meters. Eg. 4.

repeats

The number of times the mesh is placed between spline points. Eg 4.

width

Width of track mesh in meters. Eg 0.5.

Additional Tags

Kind mesh-reducing-track is derived from kind track and shares most of the same tags which are detailed in the [KIND TRACK](#) section of this chapter.

unit_mesh

The filename of the long mesh, which must be placed in a subdirectory of the same name as the mesh.

Only the file name is entered, not the directory name nor the file extension. For example, the full pathname and extension is "rockwall/rockwall.im". Enter only "rockwall" in the text input box.

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bend around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

isroad

Specifies track is a road with cars, set to 1 for cars to appear on road.

istrack

0 = This is not rail tracks.

shadows

BLAH

upright

Specifies whether the bridge "legs" point vertically, or perpendicular to the spline.

visible-on-minimap

Specifies whether the object\track is displayed on the minimap.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Track Sound, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

bendy, carrate, casts_shadows, endlength, grounded, isroad, istrack, length, repeats, rgb, shadows, upright, visible-on-minimap, width, kuid, trainz-build, category-class, category-region, category-era, username, kind, unit_mesh, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, divider, dont-flip-terminator, hidden, initiator, invisible, license, light, organisation, terminator, uncached_alphas, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

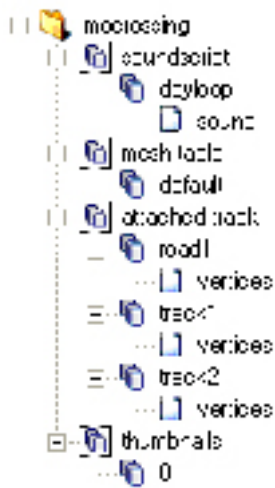
KIND: MOCROSSING

Description

Combined rail and road crossings that react to trains or script control. This allows animation, special lighting effects and attachment points for rail track and roads.

Container Structure

A well formed mocrossing kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mocrossing	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
soundscript	
dayloop	
repeat-delay	data
distance	data
sound	
0	file
mesh-table	
default	

mesh	file
anim	file
auto-create	data
attached-track	
road1	
track	kuid
useadjoiningtracktype	data
vertices	
0	data
1	data
track1	
track	kuid
useadjoiningtracktype	data
vertices	
0	data
1	data
thumbnails	
0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid
Generated automatically.
trainz-build
Automatically set to 2.5 for 2006 assets.
category-class
Should be “WX - Accessories”
See the “Wayside” section of the “Classes and Codes” appendix located at the end of this document.
category-region
See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.
category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “mocrossing”.

Additional Containers

Soundscript Container

Controls the looping sound made by the object.

Attached Track Container

Auto-generated spline track. Generated through attachment points located within the default mesh. Attached-tracks update automatically to the spline track connected to it. You may over-ride this auto-update feature by adding useadjoiningtracktype 0

Note. Correct track end attachment orientation is essential. The Y axis must point ‘out’ at the correct angle. The Z axis must point ‘up’.

The Attached Track Container has the following tags and containers:

track

Kuid of the track to be used.

useadjoiningtracktype

Indicates whether the track type should change to match that of the first track joined to the object.

Vertex

Attachment points at which to place track.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Queues Container, Smoke Container, SoundScript Container, Mesh Table, Attached Track Container, Attached Trigger Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autaname, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de,

description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, surveyor-name-label, surveyor-only, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: MOJUNCTION

Description

Junction control levers, which are attached to track junctions, include sound, and may be offset a specified distance from the track. They can be used to replace the default junction lever.

Container Structure

A well formed mojunction kind has the following container structure:



See Chapter 7 for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mojunction	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
trackside	data
mesh-table	
lever1	
mesh	file
auto-create	data
lever2	

mesh	<i>file</i>
soundscript	
toggle	
trigger	<i>data</i>
distance	<i>data</i>
nostartdelay	<i>data</i>
repeat-delay	<i>data</i>
sound	
0	<i>data</i>
thumbnails	
0	
image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “WX - Accessories”

[See the “Wayside” section of the “Classes and Codes” appendix located at the end of this document.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “mojunction”.

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Addition Containers

Soundscript Container

Controls the looping sound made by the object.

Notes

repeat-delay now has two values rather than one. When upgrading old assets, make sure there is a repeat delay for both values or the sound will loop endlessly when triggered.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Queues Container, Smoke Container, SoundScript Container, Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autoname, buffer-speed, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, speedlimit, surveyor-name-label, surveyor-only, trackmark, trackside, trigger, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

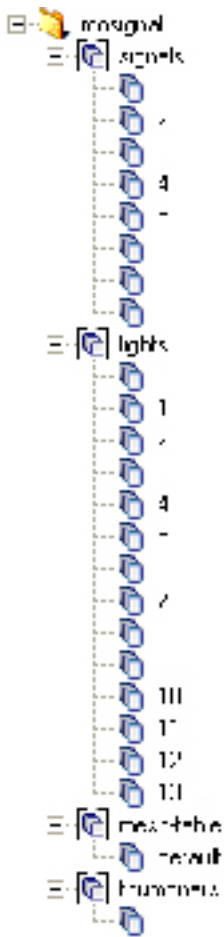
KIND: MOSIGNAL

Description

A train signal with lights (coronas). It specifies the aspects the signal is capable of displaying, the light points activated when each state is displayed, and the corona details. The signal may be offset a specified distance from the track.

Container Structure

A well formed mosignal kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mosignal

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

function	<i>data</i>
trackside	<i>data</i>
signals	
0	
light	<i>data</i>
lights	
0	
corona	<i>data</i>
mesh-table	
default	
mesh	<i>file</i>
auto-create	<i>data</i>
thumbnails	
0	
image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “WX - Accessories”

See the “Wayside” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

kind

Must be "mosignal".

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the mosignal kind also contains additional containers and tags that are specialised to the requirements of the kind.

Signal Container

When adding a signal, the user is presented with a choice of 12 separate signal containers:

Signal 1, Signal 2, Signal 3, Signal 4, Signal 5, Signal 6, Signal 7, Signal 8, Signal 9, Signal 10, Signal 11, Signal 12

The numbering of these is meaningful in that each numeric is assigned to a particular signalling state. The states are as follows:

0 STOP

1 STOP THEN PROCEED

2 CAUTION AND LEFT DIVERGE

3 CAUTION AND RIGHT DIVERGE

4 CAUTION

5 PROCEED AND LEFT DIVERGE

6 PROCEED AND RIGHT DIVERGE

7 ADVANCED CAUTION

8 PROCEED

The following two aspects are only used for scenarios....

9 SLOW

10 MEDIUM SPEED

11 [\(NO MENTION IN CCG\)](#)

12 [\(NO MENTION IN CCG\)](#)

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is light by general light value, 1 sets directional light which is affected by the position of the sun.

Lights Container

Each light point needs to have a corona associated with it. Coronas are stored in each signal object's directory alongside it's textures.

Corona

A corona is a 'glow' light effect.

Additional Tags

function

Must be set to TrackSignal.

Notes

[\(SEE PAGE 79-80 OF CCG FOR EXAMPLE TO BE INCLUDED. SEE SIGNALS.BMP IN NEWLINKS1\)](#)

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, SoundScript Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Signals, Lights.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, function, alias, author, autaname, buffer-speed, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl,

description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, speedlimit, surveyor-name-label, surveyor-only, trackmark, trackside, trigger, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

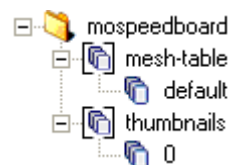
KIND: MOSPEEDBOARD

Description

A speed limit sign for Trains. It displays the maximum limit (sign texture made by the creator) and the sign may be offset a specified distance from the track. The limit to control train speed is specified in the asset in metres per second.

Container Structure

A well formed mospeedboard kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

mospeedboard

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
trackside	<i>data</i>
speedlimit	<i>data</i>

mesh-table

default

mesh	<i>file</i>
auto-create	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “WS - Trackside signage”

[See the “Wayside” section of the “Classes and Codes” appendix located at the end of this document.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

speedlimit

This value is the maximum speed allowed in meters per seconds.

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

kind

Must be “mospeedboard”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, SoundScript Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autoname, buffer-speed, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, passenger-height, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, search-limit, snapgrid, snapmode, speedlimit, surveyor-name-label, surveyor-only, trackmark, trackside, trigger, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

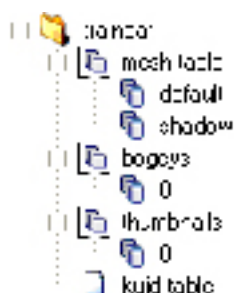
KIND: PAINTSHED-SKIN

Description

A reskin texture for a locomotive or rolling stock asset.

Container Structure

A well formed paintshed-skin kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

paintshed-reskin

origin	<i>data</i>
category-class	<i>data</i>
product-id	<i>data</i>
product-version	<i>data</i>
product-type	<i>data</i>
engine	<i>data</i>
interior	<i>kuid</i>
fonts	<i>data</i>
mass	<i>data</i>
kind	<i>data</i>

enginespec	<i>kuid</i>
enginesound	<i>kuid</i>
hornsound	<i>kuid</i>
username	<i>data</i>
description	<i>data</i>
alias	<i>kuid</i>
kuid	<i>kuid</i>
paintshed-template-used	<i>kuid</i>
paintshed-skin-used	<i>kuid</i>
category-region	<i>data</i>
category-era	<i>data</i>
trainz-build	<i>data</i>

mesh-table

default

mesh	<i>file</i>
auto-create	<i>data</i>

shadow

mesh	<i>file</i>
auto-create	<i>data</i>

bogeys

0

bogey	<i>kuid</i>
reversed	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

kuid-table

0

	<i>kuid</i>

1

	<i>kuid</i>

2

	<i>kuid</i>

3

	<i>kuid</i>

4

	<i>kuid</i>

5

	<i>kuid</i>

Specifies whether the traincar is a locomotive or rollingstock item. A loco or B-unit will have the engine tag set to true.

Mass in Kilograms, ie “7000”.

Kuid number of the interior. Inserted at a.cabfront. This tag should only be used when required (an interior is needed when the traincar is a locomotive).

References the kuid number for the traincar’s sound asset.

References the engine kuid number. This specifies the driver physics boundaries for the traincar.

References the kuid number for the traincar horn sound asset.

Kuid of the paintshed skin used. (if applicable)

Kuid of the paintshed template used. (if applicable)

Indicates how many types of numbering fonts used.

E.g.

0 = no fonts used

1 = one font

Digit textures (digit_1.tga to digit_6.tga) replaced automatically with alphanumber textures (alphanumber_0 to alphanumber_9) as numbers are changed via the Surveyor Trains tab - ‘Edit Properties’ icon (the ‘?’ icon).

2 = two fonts

Digit textures (digit_1a.tga to digit_6a.tga and digit_1b.tga to digit_6b.tga) replaced automatically with alphanumber textures (alphanumber_0a to alphanumber_9a and alphanumber_0b to alphanumber_9b) as numbers are changed via the Surveyor Trains tab - ‘Edit Properties’ icon (the ‘?’ icon).

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

description

Description of model which is used for the ‘Railyard’ information.

kind

Must be “traincar”.

alias

the kuid of the paintshed template the paintshed reskin is based on.

origin

The Country Abbreviation.

product-id

For paintshed support.

product-type

For paintshed support.

product-version

For paintshed support.

The main traincar mesh.

auto-create

Specifies that this mesh is automatically created (visible without needing to resort to a script).

shadow

The mesh model to be used as a shadow.

Bogey Container

The bogey container stores the bogeys used for the loco\rollingstock item.

The functionality of a bogey can be determined by naming it as follows:

bogey: The bogey kuid number (default for a.bog0 and a.bog1)

bogey-1: The bogey kuid number for a.bog1 (Used only if different to a.bog0)

bogey-r and bogey-1-r: Used instead of 'bogey' and bogey-1. The bogey will have reversed orientation. Note: This will cause bogey animation to play in reverse.

A Bogey container has the following tags:

reversed

Affects the direction of the bogey.

bogey

The KUID of the bogey asset.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: PAINTSHED-TEMPLATE

Description

A template for particular locomotives and rolling-stock that may be used in the integrated Paintshed utility. The template may be painted with different color schemes.

Container Structure

A well formed paintshed-template kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

paintshed-template

kind	data
paintshed-skin	kuid
kuid	kuid
username	data
category-class	data
category-era	data
trainz-build	data
category-region	data

thumbnails

0

image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be "ZX - PaintShed-Template".

See the “Train Parts” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “paintshed-template”.

Additional Tags

paintshed-skin

Kuid of the paintshed skin to be used for this template.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, paintshed-skin, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

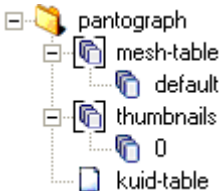
KIND: PANTOGRAPH

Description

The animated mechanisms on the roof of electric locomotives that conduct electricity from the catenary’s (wires) above. It is referenced by the pantograph tag in a traincar config file.

Container Structure

A well formed pantograph kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

pantograph

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data

mesh-table

default

mesh	file
auto-create	data
anim	file

thumbnails

0

image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “ZP - Pantographs”.

See the “Train Parts” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “pantograph”.

Mesh Table

anim

The “anim.kin” animation file for the pantograph animation.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Mesh Table.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

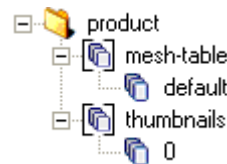
KIND: PRODUCT

Description

An individual product (commodity) that Trainz compatible rolling stock and industry assets are able to process. It specifies the type, unit of measurement and the picture icon that displays the product in the simulator. Produce and materials are product examples.

Container Structure

A well formed product kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

product

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
allows-mixing	data
instance-type	data
mass	data
product-category	kuid
icon-texture	file
product-texture	file

mesh-table

default

mesh	file
auto-create	data

thumbnails

0

image	file

width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

"I - Product", "IC - Container Category", "IP - Passenger Category", "IB - Bulkload Category", "IL - Liquid Category"

See the "Motive Power" section of the "Classes and Codes" appendix located at the end of this document.

category-region

See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.

category-era

See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be "product".

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Tags

allows-mixing

Products with this tag may be combined in a single queue along with other products of the same category.

mass

The physical mass of the product

For Containers and Passengers this is calculated in kilograms/unit, while for Liquid and Bulk loads this is calculated in kilograms/litre.

icon-texture

the in-game representation of the product when specifying the load type for a compatible rollingstock item (driver) Should be a 64x64 TGA.

product-texture

The texture to be used with load 'texture-replacement'.
ie When a hopper loads woodchips instead of it's default load of coal.

See the texture replacement section in Chapter 8 for more information.

product-category

kuid of applicable category for this product

instance-type

No definition could be found for this.

resource = Used when there is no mesh, or one only mesh is referenced in the mesh table (ie Liquids, Bulk loads etc).

instance = Used when more than one mesh is in the mesh table ie: Passengers, General Goods. 200 max.'size' per Asset.

unique = SO UNIQUE NOBODY KNOWS ANYTHING ABOUT IT :P

Notes

IN-GAME VISUALISATION OF PRODUCTS.

In TRS, products can be displayed a few ways:

1) An animated load representation.

This technique is used for bulk-category loads such as coal or woodchip products both in industry and rollingstock assets and for liquid loads through indicators adjacent to storage tanks. The animation is non-looping. Say we have an industry bulk load animation with the frames running from 0 to 30. Empty will be at frame 0 and full will be at frame 30. Texture swapping is possible for some rollingstock bulk loading assets.

Details of how to texture swapping is set up is available in the following downloadable file: http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_Load_Texture_Replacement.zip

2) Mesh attachment representation.

This technique is used for container-category loads such. 20ft and 40ft Containers, General Goods, Lumber and Logs all use this technique. If a piece of rollingstock has the potential to carry several product types (such as a flat car), it is possible to set up the loads to be mutually exclusive through it's config. That is if it has capacity of one load, it cannot load any other product types.

3) 'View details' Driver information window display.

This (of course) can be used for all rollingstock items, but specifically, it is the means to see the load of rollingstock that cannot otherwise visually display it's load. Ie. Tank Cars and enclosed Box Cars.

Box cars can be setup to take General Goods but without load attachments.

Note: Tank cars and tenders may use a separate animated 'loader' mesh to visualise the loading of liquids. This is set up through the industry asset's script and the rollingstock item's config. For script reference please refer to index.chm found in \Trainz\scripts\docs directory.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, allows-mixing, instance-type, mass, product-category, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, icon-texture, icon0, icon1, icon2, icon3, license, organisation, preview-mesh-kuid, preview-scale, product-texture, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: PRODUCT-CATEGORY

Description

A category class of products (commodities) that Trainz compatible rolling-stock and industry assets are able to process. It specifies the type, unit of measurement and the picture icon that displays the category on Surveyor or Driver. Bulk, liquid, passengers and containers are product category examples.

Container Structure

A well formed product-category kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

product-category

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following:

"I - Product", "IC - Container Category", "IP - Passenger Category", "IB - Bulkload Category", "IL - Liquid Category"

[See the "Product" section of the "Classes and Codes" appendix located at the end of this document.](#)

category-region

[See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be "Product-Category".

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: PROFILE

Description

A Profile is known as a Session in Trainz. This kind creates a session defining a single route with different consists, starting points, and industry outputs. Different sets of trains may be used in each different session.

Container Structure

A well formed profile kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

profile

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
map-kuid	<i>kuid</i>

info-page *file*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be "YP - Profile/Session".

[See the "Maps & Scenarios" section of the "Classes and Codes" appendix located at the end of this document.](#)

category-region

[See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor and driver menus.

kind

Must be "profile".

Additional Tags

map-kuid

Kuid of the map attached to this session.

info-page

Filename of the HTML information page for the session.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String

Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, map-kuid, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, info-page, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: REGION

Description

A region is chosen in Surveyor to create a new map or route. This Kind creates a new region in addition to the in-built regions such as Australia or USA for example. The region can define geographical location, road traffic and weather conditions.

Container Structure

A well formed region kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

region

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
watercolor	<i>data</i>
defaultjunction	<i>data</i>
ontheright	<i>data</i>
longitude	<i>data</i>
latitude	<i>data</i>

altitude	<i>data</i>
car0	<i>kuid</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “Traincar”.

Additional Tags

watercolor

RGB colour value of the water for the region.

defaultjunction

Default type of junction in this region.

ontheright

Cars drive on the right side of the road.

longitude

longitude of this region

latitude

latitude of this region

altitude

altitude of this region

car0, car1, car2, car3, car4, car5, car6, car7, car8, car9,car10, car11, car12, car13, car14, car15

Each of these tags storas the kuid of a car to be used on the roads.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, watercolor, defaultjunction, ontheright, longitude, latitude, altitude, author, car0, car1, car10, car11, car12, car13, car14, car15, car2, car3, car4, car5, car6, car7, car8, car9, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: SCENERY

Description

A basic scenery asset that supports night lighting, smoke (particle) effects, sound and animation. It is height adjustable and forms the majority of map objects used.

Container Structure

A well formed scenery kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

scenery

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

mesh-table

default

mesh	<i>file</i>
auto-create	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

See the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “scenery”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Tags

backdrop

Specifies whether the object is treated as a backdrop or not. (stays visible even when far from the camera)

random-color-high-hsb

For clutter-mesh objects, specifies a color range for tinting purposes. HSB color space.

random-color-low-hsb

For clutter-mesh objects, specifies a color range for tinting purposes. HSB color space.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, Soundscript Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, autaname, backdrop, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, preview-mesh-kuid, preview-scale, random-color-high-hsb, random-color-low-hsb, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, snapmode, surveyor-name-label, surveyor-only, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

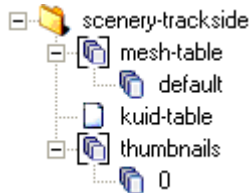
KIND: SCENERY-TRACKSIDE

Description

A special scenery asset attached to rail track with the offset distance from the track specified in the asset. Examples could include a signal box, or dummy track sign or track object.

Container Structure

A well formed scenery-trackside kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

scenery-trackside

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
trackside	data
preview-mesh-kuid	kuid

kuid-table

0	kuid

mesh-table

default

mesh	file
auto-create	data

thumbnails

0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

See the “Classes and Codes” appendix located at the end of this document to select an appropriate class.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “scenery”.

Mesh Table

Default

Contains the default mesh. Auto-create should be set to true in order to make the mesh visible.

Additional Tags

collate-meshes

Enables clutter-mesh support (eg. fast trees)

buffer-speed

Used for buffers; specifies the maximum speed up to which the buffer will stop a train.

speedlimit

This value is the maximum speed allowed in meters per seconds.

preview-mesh-kuid

the mesh to be used in the surveyor preview area. This is useful when an asset has a large bounding box. Ie the “Airport” with it’s jet animation.

trackmark

Specifies that the object is a trackmark.

trackside

This is a value that is the distance in meters the object is placed relative to the center of the track. Negative values will put the object on the left side of the track, and positive values will appear on the right.

trigger

Specifies that the object is a trigger.

search-limit

Not required. For internal use only.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, Soundscript Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, trackside, alias, author, autoname, backdrop, buffer-speed, category-keyword, class, collate-meshes, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, height-range, icon-texture, icon0, icon1, icon2, icon3, license, light, nightmode, organisation, preview-mesh-kuid, preview-scale, random-color-high-hsb, random-color-low-hsb, rgb, rollstep, rotate, rotate-xyz-range, rotstep, script, search-limit, snapgrid, snapmode, speedlimit, surveyor-name-label, surveyor-only, trackmark, trigger, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: STEAM-ENGINE

Description

The special engine specification for steam locomotives which defines the detailed performance requirements such as throttle requirements, engine and braking performance, boiler capacity and steam attributes.

Container Structure

A well formed steam-engine kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

steam-engine

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data

flowsize

trainbrakepipe	data
epreservoirpipe	data
no3pipe	data
no4pipe	data
auxreservoirvent	data
auxreservoir_no3	data
auxreservoir_trainbrakepipe	data
autobrakecylindervent	data

auxreservoir_autobrakecylinder	data
equaliser_mainreservoir	data
equaliservent	data
equaliserventhandleoff	data
equaliserventemergency	data
no3pipevent	data
no3pipe_mainreservoir	data
compressor	data
trainbrakepipe_reservoir	data
trainbrakepipevent	data
no3pipe_autobrakecylinder	data
epreservoirpipe_autobrakecylinder	data
mainreservoir_ep	data
vacuumbrakepipe	data
vacuumbrakepipereleasevent	data
vacuumbrakepipevent	data
vacuumbrakereservoir_vacuumbrakepipe	data
vacuumbrakecylinder_vacuumbrakepipe	data
highspeedexhauster_vacuumbrakepipe	data
volume	
scale	data
trainbrakepipe	data
epreservoirpipe	data
no3pipe	data
no4pipe	data
auxreservoir	data
autobrakecylinder	data
vacuumbrakepipe	data
vacuumbrakereservoir	data
vacuumbrakecylinder	data
mainreservoir	data
equaliser	data
independantbrakecylinder	data
pressure	
scale	data

compressor	<i>data</i>		
mainreservoir	<i>data</i>	steam	
highspeedexhauster	<i>data</i>	firebox-to-boiler-heat-flow	<i>data</i>
brakepipe	<i>data</i>	firebox-efficiency	<i>data</i>
brakeinitial	<i>data</i>	boiler-volume	<i>data</i>
brakefull	<i>data</i>	minimum-volume	<i>data</i>
indbrakefull	<i>data</i>	maximum-volume	<i>data</i>
trainbrakepipe_start	<i>data</i>	initial-boiler-temperature	<i>data</i>
epreservoirpipe_start	<i>data</i>	water-injector-rate	<i>data</i>
no3pipe_start	<i>data</i>	piston-volume-min	<i>data</i>
no4pipe_start	<i>data</i>	piston-volume-max	<i>data</i>
auxreservoir_start	<i>data</i>	piston-area	<i>data</i>
autobrakecylinder_start	<i>data</i>	piston-angular-offsets	<i>data</i>
vacuumbrakepipe_start	<i>data</i>	firebox-to-boiler-heat-flow-idle	<i>data</i>
vacuumbrakereservoir_start	<i>data</i>	burn-rate-idle	<i>data</i>
vacuumbrakecylinder_start	<i>data</i>	boiler-to-piston-flow	<i>data</i>
mainreservoir_start	<i>data</i>	piston-to-atmosphere-flow	<i>data</i>
equaliser_start	<i>data</i>	safety-valve-low-pressure	<i>data</i>
independantbrakecylinder_start	<i>data</i>	safety-valve-low-flow	<i>data</i>
mass		safety-valve-high-pressure	<i>data</i>
scale	<i>data</i>	safety-valve-high-flow	<i>data</i>
fuel	<i>data</i>	max-fire-coal-mass	<i>data</i>
motor		max-fire-temperature	<i>data</i>
resistance	<i>data</i>	shovel-coal-mass	<i>data</i>
adhesion	<i>data</i>	burn-rate	<i>data</i>
maxvoltage	<i>data</i>	fuel-energy	<i>data</i>
maxspeed	<i>data</i>	firebox-volume	<i>data</i>
brakeratio	<i>data</i>	main-reservoir-volume	<i>data</i>
max-accel	<i>data</i>	westinghouse-volume	<i>data</i>
max-decel	<i>data</i>	thumbnails	
axle-count	<i>data</i>	0	
surface-area	<i>data</i>	image	<i>file</i>
moving-friction-coefficient	<i>data</i>	width	<i>data</i>
air-drag-coefficient	<i>data</i>	height	<i>data</i>
throttle-notches	<i>data</i>	TAGS AND CONTAINERS	

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following:

, “AS - Steam Loco & Tender”, “AT - Steam Tank”.

See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “steam-engine”.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the steam-engine kind also contains additional containers and tags that are specialised to the requirements of the kind.

Steam Container

The Steam Container stores specialised variables related to steam engines.

The Steam Containers has the following tags:

boiler-to-piston-flow

Relative energy

boiler-volume

Physical volume of boiler in Litres.

burn-rate

Coal consumption rate.

burn-rate-idle

Coal consumption rate when idle.

firebox-efficiency

Atmosphere leakage. 1.0 = No leakage.

firebox-to-boiler-heat-flow

Rate of heat flow from firebox to boiler and viseversa.

firebox-to-boiler-heat-flow-idle

Rate of heat flow from firebox to boiler when idle.

firebox-volume

Physical volume of firebox in Litres.

fuel-energy

Relative energy in kilojoules per kilogram of coal

main-reservoir-volume

Main reservoir volume in litres.

max-fire-coal-mass

The maximum mass of coal the firebox can take in kilograms.

max-fire-temperature

Maximum heat obtainable in degrees K. (Kelvin scale temperature)

piston-angular-offsets

Leave this setting

piston-volume-min

The volume of the space in the cylinder ahead of the piston at the end of a full stroke.

piston-volume-max

The volume of the space in the cylinder ahead of the piston at the end of a full stroke.

piston-area

The cross section of one piston in m2. It is assumed there is one piston only on each side of the locomotive.

piston-to-atmosphere-flow

Atmospheric leakage from piston. Nominal hole size.

safety-valve-low-pressure

When boiler pressure hits this value in kPa the safety-valve-low-flow release is initiated.

safety-valve-low-flow

Lower pressure valve release. Nominal hole size.

safety-valve-high-pressure

When boiler pressure hits this value in kPa the safety-valve-low-flow release is initiated.

safety-valve-high-flow

Higher pressure valve release. Nominal hole size.

shovel-coal-mass

Amount of coal in one shovel load in kg

water-injector-rate

Water injection rate into boiler in Litres/second

westinghouse-volume

Westinghouse volume in Litres.

initial-boiler-temperature

Boiler temperature on loading the game.

max-coal-mass

Maximum volume of coal.

maximum-volume

The maximum volume of the piston(in litres)

minimum-volume

The minimum volume of the piston(in litres)

Mass Container

The mass container stores information related to fuel consumption. These tags aren't in use and shouldn't generally be used.

The mass container has the following tags:

scale

Multiplies fuel mass by given value, not currently in use, generally leave this setting.

fuel

Fuel level, not currently in use, generally leave this setting.

Motor Container

The Motor Container stores an assortment of values related to motor function, particularly that of DCC.

resistance

Power figure for DCC, higher resistance value=less power

adhesion

Adhesion parameter, higher value=greater adhesion

maxvoltage

Generally leave this setting

maxspeed

Maximum speed for DCC, expressed in metres per second.

brakeratio

Brake force for pressure reduction.

max-accel

Parameters for DCC acceleration & deceleration.

max-decel

Parameters for DCC acceleration & deceleration.

throttle-notches

Number of throttle notches.

axle-count

Resistance - Axle Count.

surface-area

Resistance - Surface Area.

moving-friction-coefficient

Resistance - Moving friction.

air-drag-coefficient

Resistance - Air drag.

Flowsize Container

Flowsize settings specify the rate of flow through the pipes. Generally these setting should be left unaltered.

The Flowsize Container has the following tags:

trainbrakepipe

Flowsize of the brake pipe.

epreservoirpipe

Flowsize of the electric pneumatic braking

no3pipe

Flowsize of the independent brake pipe.

no4pipe

Flowsize of the bail pipe.

auxreservoirvent

Flowsize of the auxiliary reservoir vent.

auxreservoir_no3

Flowsize of the auxiliary independent brake pipe.

auxreservoir_trainbrakepipe

Flowsize of the auxiliary reservoir brake pipe.

autobrakecylindervent

Flowsize of the automatic brake cylinder vent.

auxreservoir_autobrakecylinder

Flowsize of the auxiliary reservoir automatic brake cylinder.

equaliser_mainreservoir

Flowsize of the equaliser main reservoir.

equaliservent

Flowsize of the equaliser vent.

equaliserventhandleoff

Flowsize of the equaliser to the atmosphere when in the "handle off" position.

equaliserventemergency

Flowsize of the emergency equaliser vent.

no3pipevent

Flowsize of the independent brake pipe.

no3pipe_mainreservoir

Flowsize of the independent brake main reservoir.

compressor

Flowsize of the compressor.

trainbrakepipe_reservoir

Flowsize of the brake pipe reservoir.

trainbrakepipevent

Flowsize of the brake pipe vent.

no3pipe_autobrakecylinder

Flowsize of the independent automatic brake pipe cylinder.

epreservoirpipe-autobrakecylinder

Flowsize of the electro pneumatic automatic brake cylinder reservoir.

mainreservoir_ep

Flowsize of the electro pneumatic main reservoir.

vacuumbrakepipe

Flowsize of the vacuum brake pipe.

vacuumbrakepipereleasevent

Flowsize of the vacuum brake pipe release vent.

vacuumbrakepipevent

Flowsize of the vacuum brake pipe vent.

vacuumbrakereservoir_vacuumbrakepipe

Flowsize of the vacuum brake pipe reservoir.

vacuumbrakecylinder_vacuumbrakepipe

Flowsize of the vacuum brake pipe cylinder.

highspeedexhauster_vacuumbrakepipe

Flowsize of the high speed exhauser vacuum brake pipe.

Volume Container

The volume container stores information regarding the size of pipes and appliances. Generally these settings should remain unaltered.

The Volume Container has the following tags:

scale

multiplies volume by given value, generally leave this setting.

trainbrakepipe

brake pipe volume

epreservoirpipe

For electro pneumatic braking - not currently in use, generally leave this setting.

no3pipe

Independent brake pipe

no4pipe

Bail pipe - not currently in use, generally leave this setting.

auxreservoir

Auxiliary reservoir volume.

autobrakecylinder

Brake cylinder volume.

vacuumbrakepipe

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakereservoir

For vacuum braking - not currently in use, generally leave this setting.

vacuumbrakecylinder

For vacuum braking - not currently in use, generally leave this setting.

mainreservoir

Main reservoir volume.

equaliser

Equalising reservoir volume

independantbrakecylinder

Loco brake cylinder volume

Steam-Power Coordinate

This container is a graph, similar to the throttle and dynamic-brake graphs. Velocity incoming, Power multiplier outgoing.

Coordinate Element

[Find out how these work somehow.](#)

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Throttle Power Container, Dynamic Brake Container, Pressure Container, Mass Container, Motor Container, Flowsize Container, Volume Container, Steam container, Steam-Power.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: TEXTURE

Description

A simple texture asset that can be referenced from another asset (for example a custom corona) by referencing its kuid.

Container Structure

A well formed texture kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

texture

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
texture	<i>file</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following values: "J - Texture", "JC - Corona", "JI - Icon", "JP - Particle Effect Texture", "JO

- Other Texture”.

See the “Texture” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “texture”.

texture

The texture file to be used.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, texture, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: TEXTURE-GROUP

Description

Defines a group of textures as an asset that can be referenced from another asset or via scripting.

Container Structure

A well formed texture-group kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

texture-group

kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data

textures

0	file
1	file
2	file

thumbnails

0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be one of the following values: “J - Texture”, “JC - Corona”, “JI - Icon”, “JP - Particle Effect Texture”, “JO - Other Texture”.

See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “texture-group”.

Additional Containers

As well as containing all of the common tags and containers detailed in Chapter 5 , the texture-group kind also contains additional containers and tags that are specialised to the requirements of the kind.

Textures Container

The textures container stores a list of additional textures to be used in the texture group.

Texture

Texture path.

Notes

The ONLY use for this asset is for reference by the script function “SetFXTextureReplacement”.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Textures Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, texture, author, category-

keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

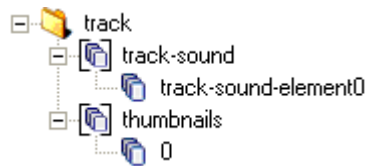
KIND: TRACK

Description

Variable length spline based track, roads, and other scenery items. Tracks may include initiator and terminator segments, and are height adjustable. Other uses for this kind include fences, power lines and hedges.

Container Structure

A well formed track kind has the following container structure:



See Chapter 7 for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

track	
bendy	data
carrate	data
casts_shadows	data
endlength	data
grounded	data
isroad	data
istrack	data
length	data
repeats	data
rgb	data
shadows	data
upright	data
visible-on-minimap	data
width	data
kuid	kuid
trainz-build	data

category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
bridgetrack	<i>kuid</i>
height	<i>data</i>
trackoffsets	<i>data</i>
kuid-table	
0	<i>kuid</i>
thumbnails	
0	
image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “TR - Rails”.

See the “Track” section of the “Classes and Codes” appendix located at the end of this document.

category-region

See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.

category-era

See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “track”.

rgb

This value should be left as default.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the track kind also contains additional containers and tags that are specialised to the requirements of the kind.

Additional Tags

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bend around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

grounded

Height in meters for the road to be offset from terrain.

isroad

Specifies track is a road with cars, set to 1 for cars to appear on road.

istrack

0 = This is not rail tracks.

length

Length of track segment in meters

repeats

The number of times the mesh is placed between spline points

shadows

BLAH

upright

Specifies whether the bridge “legs” point vertically, or perpendicular to the spline.

visible-on-minimap

Specifies whether the object\track is displayed on the minimap.

width

Width of track mesh in meters.

bridgetrack

Kuid for the type of rail or road used on bridge.

height

Height from the track level to the base. Should be negative.

trackoffsets

Distance in meters the rail/s are attached to the center of the spline. Any number of tracks can be attached to the spline, only splines with the same track offsets can be connected together.

divider

Name of the model to use as the middle bridge section. Placed in subfolder with same name.

dont-flip-terminator

Terminator model isn't mirrored on one side.

hidden

Prevents the spline from being rendered.

initiator

Name of model to use at start of bridge, placed in subfolder with same name.

invisible

Specifies whether the object is invisible.

terminator

Name of model to use at end of bridge, placed in subfolder with same name.

uncached_alphas

This is used in certain situations to improve alpha sorting. This should only be set to 1 for tracks that use an alpha texture and are always placed flat near the ground.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Track Sound, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String

Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

bendy, carrate, casts_shadows, endlength, grounded, isroad, istrack, length, repeats, rgb, shadows, upright, visible-on-minimap, width, kuid, trainz-build, category-class, category-region, category-era, username, kind, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, divider, dont-flip-terminator, hidden, initiator, invisible, license, light, organisation, terminator, uncached_alphas, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

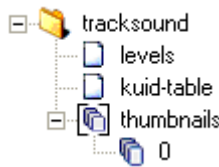
KIND: TRACKSOUND

Description

A sound asset that is referenced by track or bogeys to play a different sound from the default track/train sound (for example when a train travels over a bridge or through a tunnel).

Container Structure

A well formed tracksound kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

tracksound	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
min-distance	data
max-distance	data
levels	

0 *data*

1 *data*

thumbnails

0

image *file*

width *data*

height *data*

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Value should be “XSE - Unclassified”.

[See the “Classes and Codes” appendix located at the end of this document for more information.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. Though this asset doesn’t appear in the Surveyor menu, the username is used to identify the asset in the “Content Manager Plus” and “Content Creator Plus” programs.

kind

Must be “tracksound”.

Additional Containers

As well as containing all of the common tags and containers detailed in [Chapter 5](#), the tracksound kind also contains additional containers and tags that are specialised to the requirements of the kind.

Levels Container

Relative sound levels. The sound is silent until 0.1 m/s, ramping up in volume until 10.0 m/s, constant maximum after that. Note, a value below 0.1 will not play a sound.

An example Levels setting would be:

0 0.1

1 10

Multiple sound files may be used, idle 1.wav, idle 2.wav for instance.

Additional Tags

min-distance

Minimum distance at which the sound is played. Eg 0.

max-distance

Maximum distance at which the sound is played. Eg 10000.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Levels Container.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, min-distance, max-distance, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: TRAINCAR

Description

A locomotive or rolling stock item. A traincar specifies the dependant assets (bogey, engine sound, engine specification, pantograph and interior), to make a complete asset.

Container Structure

A well formed traincar kind has the following container structure:



A TYPICAL TRAINCAR KIND MAY HAVE THE FOLLOWING TAGS:

traincar	
kuid	kuid
trainz-build	data
category-class	data
category-region	data
category-era	data
username	data
kind	data
engine	kuid
mass	data
description	data
enginesound	kuid
hornsound	kuid
enginespec	kuid
pantograph	kuid
company	data
interior	kuid
mesh-table	
default	
mesh	file

auto-create	data
shadow	
mesh	data
bogeys	
bogey	
reversed	data
bogey	kuid
thumbnails	
0	
image	file
width	data
height	data

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid	Generated automatically.
trainz-build	Automatically set to 2.5 for 2006 assets.
category-class	See the “Motive Power” section of the “Classes and Codes” appendix located at the end of this document.
category-region	See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.
category-era	See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.
username	Asset username. This will be the name that appears in the surveyor menu.
kind	Must be “Traincar”.
engine	Specifies whether the traincar is a locomotive or rollingstock item. A loco or B-unit will have the engine tag set to true.

mass

Mass in Kilograms, ie “7000”.

description

Description of model which is used for the ‘Railyard’ information.

enginesound

References the kuid number for the traincar’s sound asset.

hornsound

References the kuid number for the traincar horn sound asset.

enginespec

References the engine kuid number. This specifies the driver physics boundaries for the traincar.

pantograph

The pantograph kuid number inserted at a.pant0, a.pant1, etc. Use this tag only when needed.

company

The Locomotive or car owner, eg “QR”.

interior

Kuid number of the interior. Inserted at a.cabfront. This tag should only be used when required (an interior is needed when the traincar is a locomotive).

Mesh Table

default

The main traincar mesh.

auto-create

Specifies that this mesh is automatically created (visible without needing to resort to a script).

shadow

The mesh model to be used as a shadow.

Bogey Container

The bogey container stores the bogeys used for the loco\rollingstock item.

The functionality of a bogey can be determined by naming it as follows:

bogey: The bogey kuid number (default for a.bog0 and a.bog1)

bogey-1: The bogey kuid number for a.bog1 (Used only if different to a.bog0)

bogey-r and bogey-1-r: Used instead of ‘bogey’ and bogey-1. The bogey will have reversed orientation. Note: This will cause bogey animation to play in reverse.

A Bogey container has the following tags:

reversed

Affects the direction of the bogey.

bogey

The KUID of the bogey asset.

See [Chapter 7](#) for an example asset of this kind.

Additional Tags

cabinsway

Cabin sway multiplier. Eg -2.

backlength

Obsolete.

backpivot

Obsolete.

disable-extra-track-sounds

Disables the “click-clack” tracksounds. (0, 1)

ditch_color

RGB ditch light colour. Eg. 255,255,255.

fonts

Indicates how many types of numbering fonts used.

E.g.

0 = no fonts used

1 = one font

Digit textures (digit_1.tga to digit_6.tga) replaced automatically with alphanumeric textures (alphanumeric_0 to alphanumeric_9) as numbers are changed via the Surveyor Trains tab - ‘Edit Properties’ icon (the ‘?’ icon).

2 = two fonts

Digit textures (digit_1a.tga to digit_6a.tga and digit_1b.tga to digit_6b.tga) replaced automatically with alphanumeric textures (alphanumeric_0a to alphanumeric_9a and alphanumeric_0b to alphanumeric_9b) as numbers are changed via the Surveyor Trains tab - ‘Edit Properties’ icon (the ‘?’ icon).

fonts-path

Replaces asset-filename usage for ‘fonts’.

frontlength

Obsolete.

frontpivot

Obsolete.

light_color

RGB headlight colour. Eg. 255,255,255.

max-coupler-gap

maximum gap expected between couplers of this type (meters).

origin

The Country Abbreviation.

paintshed-skin-used

Kuid of the paintshed skin used. (if applicable)

paintshed-template-used

Kuid of the paintshed template used. (if applicable)

product-id

For paintshed support.

product-type

For paintshed support.

product-version

For paintshed support.

smoke_fastlife

Longevity of smoke particles at normal speed.

smoke_height

How hard particles are pushed out of the stack.

smoke_random

Level of particle excitation.

smoke_shade

Smoke opacity. (0 - 1)

smoke_slowlife

Longevity of smoke particles at low speed.

tender

Specifies that the traincar is a tender.

Notes

[See Chapter 7 for some examples using the traincar kind.](#)

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Queues Container, Smoke Container, SoundScript Container, Mesh Table, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions, Bogeys.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, engine, mass, alias, author, autaname, backlength, backpivot, buffer-speed, cabinsway, category-keyword, class, company, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, disable-extra-track-sounds, ditch_color, enginesound, enginespec, floating, fonts, fonts-path, frontlength, frontpivot, height-range, hornsound, icon-texture, icon0, icon1, icon2, icon3, interior, license, light, light_color, max-coupler-gap, nightmode, organisation, origin, paintshed-skin-used, paintshed-template-used, pantograph, preview-mesh-kuid, preview-scale, product-id, product-type, product-version, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, search-limit, smoke_fastlife, smoke_fastspeed, smoke_height, smoke_random, smoke_shade, smoke_slowlife, snapgrid, snapmode, speedlimit, surveyor-name-label, surveyor-only, tender, trackmark, trackside, trigger, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: TUNNEL

Description

Road and rail tunnel variable length splines. These allow the spline to be placed below ground and usually require an integrated initiator and termination mesh as a tunnel entrance.

Container Structure

A well formed tunnel kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

tunnel

bendy	<i>data</i>
carrate	<i>data</i>
casts_shadows	<i>data</i>
endlength	<i>data</i>
grounded	<i>data</i>
isroad	<i>data</i>
istrack	<i>data</i>
length	<i>data</i>
repeats	<i>data</i>
rgb	<i>data</i>
shadows	<i>data</i>
upright	<i>data</i>
visible-on-minimap	<i>data</i>
width	<i>data</i>
kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>

kind	<i>data</i>
bridgetrack	<i>kuid</i>
height	<i>data</i>
trackoffsets	<i>data</i>
initiator	<i>data</i>
divider	<i>data</i>
terminator	<i>data</i>
kuid-table	
0	<i>kuid</i>
thumbnails	
0	
image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be “TT - Tunnel”.

[See the “Track” section of the “Classes and Codes” appendix located at the end of this document.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “bridge”.

rgb

This value should be left as default.

height

Height from the track level to the base. Must be a positive value in order to place the tunnel under the ground.

bridgetrack

Kuid of the track type to be used.

Kuid Table

The kuid of the track\road used in the asset should be present here, as should those of any other referenced assets.

Additional Tags

bendy

Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bend around corners.

carrate

Defines traffic density on road (minimum seconds between each car generated). 0 = No traffic. Number must be greater than 3.

casts_shadows

Toggles whether the shadow model is displayed.

endlength

Length in meters of the initiator and terminator models.

grounded

Height in meters for the road to be offset from terrain.

isroad

Specifies track is a road with cars, set to 1 for cars to appear on road.

istrack

0 = This is not rail tracks.

length

Length of track segment in meters

repeats

The number of times the mesh is placed between spline points

shadows

BLAH

upright

Specifies whether the bridge “legs” point vertically, or perpendicular to the spline.

visible-on-minimap

Specifies whether the object\track is displayed on the minimap.

width

Width of track mesh in meters.

bridgetrack

Kuid for the type of rail or road used on bridge.

height

Height from the track level to the base. Should be negative.

trackoffsets

Distance in meters the rail/s are attached to the center of the spline. Any number of tracks can be attached to the spline, only splines with the same track offsets can be connected together.

divider

Name of the model to use as the middle bridge section. Placed in subfolder with same name.

dont-flip-terminator

Terminator model isn't mirrored on one side.

hidden

Prevents the spline from being rendered.

initiator

Name of model to use at start of bridge, placed in subfolder with same name.

invisible

Specifies whether the object is invisible.

terminator

Name of model to use at end of bridge, placed in subfolder with same name.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Track Sound, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

bendy, carrate, casts_shadows, endlength, grounded, isroad, istrack, length, repeats, rgb, shadows, upright, visible-on-minimap, width, kuid, trainz-build, category-class, category-region, category-era, username, kind, bridgetrack, height, trackoffsets, alias, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, divider, dont-flip-terminator, hidden, initiator, invisible, license, light, organisation, terminator, uncached_alphas, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

KIND: TURNTABLE

Description

A turntable asset for moving or rotating traincars, specifying the static and moving part of the turntable. Animated rotation (turntable) and lateral translation (transfer table) assets are supported.

Container Structure

A well formed turntable kind has the following container structure:

See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

turntable

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>
snapmode	<i>data</i>
dighole	<i>data</i>
light	<i>data</i>
angle	<i>data</i>
looping	<i>data</i>

mesh-table

default

mesh	<i>file</i>

auto-create *data*

turntable

mesh *file*

auto-create *data*

attached-track

track_turntable

track *kuid*

useadjoiningtracktype *data*

vertices

0 *data*

1 *data*

track0_base

track *kuid*

useadjoiningtracktype *data*

vertices

0 *data*

1 *data*

track1_base

track *data*

useadjoiningtracktype *data*

vertices

0 *data*

1 *data*

track2_base

track *kuid*

useadjoiningtracktype *data*

vertices

0 *data*

1 *data*

track3_base

track *kuid*

useadjoiningtracktype *data*

vertices

0 *data*

1 *data*

track4_base

track	<i>kuid</i>
useadjoiningtracktype	<i>data</i>

vertices

0	<i>data</i>
1	<i>data</i>

track10_base

track	<i>kuid</i>
useadjoiningtracktype	<i>data</i>

vertices

0	<i>data</i>
1	<i>data</i>

kuid-table

0	<i>kuid</i>

thumbnails

0	
image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be set to "BR - Railway (scenery non-functional)".

See the "Classes and Codes" appendix located at the end of this document for more information.

category-region

See the "Region Codes" appendix located at the end of this document for a list of valid category-region values.

category-era

See the "Era Codes" appendix located at the end of this document for a list of valid category-era values.

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be "turntable".

Additional Tags

angle

Specifies the angles at which the turntable stops. Not used if the turntable is set up as animation.

frame-rate

Generally make this 30 (frames per second)

keyframes

Specifies where on the animation the turntable is to stop.

looping

specifies that the turntable can go all the way around, rather than stopping at a certain point.

snapmode

Specifies the alignment of the turntable to the surveyor grid. 1 = origin snaps to grid (use for removing even dighole values), 2 = origin snaps to the center of a grid square (use for odd dighole values).

dighole

Specifies the number of grid segments (length, width) to be removed from the surveyor grid to accommodate the turntable pit.

light

Sets lighting to be used for object to be ambient or directional. 0 sets ambient lighting and object is lit by general light value (uniform colouring), 1 sets directional light which is affected by the position of the sun (shows shadows on the object surfaces).

Notes

ANIMATED TURNABLES

Turntables can be set up as an animation.

Keyframes can be specified as the stopping points much like 'angles' above. Use attached-tracks at keyframe points. See Test Asset example:<http://www.auran.com/TRS2004/downloads/contentcreation/TransporterTestAsset.zip>

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

Mesh Table, Queues Container, Smoke Container, Soundscript Container, Attached Track Container, Attached Trigger Container, String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, snapmode, alias, angle, author, autaname, category-keyword, class, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, dighole, floating, frame-rate, height-range, icon-texture, icon0, icon1, icon2, icon3, keyframes, license, light, looping, nightmode, organisation, preview-mesh-kuid, preview-scale, rgb, rollstep, rotate, rotate-yz-range, rotstep, script, snapgrid, surveyor-name-label, surveyor-only, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

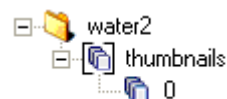
KIND: WATER2

Description

Animated water texture assets.

Container Structure

A well formed water2 kind has the following container structure:



See [Chapter 7](#) for an example asset of this kind.

A TYPICAL ASSET OF THIS KIND MAY HAVE THE FOLLOWING TAGS:

water2

kuid	<i>kuid</i>
trainz-build	<i>data</i>
category-class	<i>data</i>
category-region	<i>data</i>
category-era	<i>data</i>
username	<i>data</i>
kind	<i>data</i>

thumbnails

0

image	<i>file</i>
width	<i>data</i>
height	<i>data</i>

TAGS AND CONTAINERS

The following tags are further defined in Chapter 5:

kuid

Generated automatically.

trainz-build

Automatically set to 2.5 for 2006 assets.

category-class

Should be set to “EW - Water”.

[See the “Classes and Codes” appendix located at the end of this document for more information.](#)

category-region

[See the “Region Codes” appendix located at the end of this document for a list of valid category-region values.](#)

category-era

[See the “Era Codes” appendix located at the end of this document for a list of valid category-era values.](#)

username

Asset username. This will be the name that appears in the surveyor menu.

kind

Must be “water2”.

THIS KIND HAS THE FOLLOWING:

CONTAINERS:

String Table, Chinese String Table, Czech String Table, French String Table, German String Table, Italian String Table, Polish String Table, Russian String Table, Spanish String Table, KUID Table, Obsolete Table, Thumbnails, Privileges, Extensions.

TAGS:

kuid, trainz-build, category-class, category-region, category-era, username, kind, author, category-keyword, contact-email, contact-website, description, description-cn, description-cz, description-de, description-es, description-fr, description-it, description-pl, description-ru, license, organisation, username-cn, username-cz, username-de, username-es, username-fr, username-it, username-pl, username-ru.

CHAPTER 7

Kind Examples

The purpose of this chapter is to Define all the Kinds that are included in TRS2006, and to describe the Container and Tags that are applicable to each Kind. Please refer to [Chapter 6](#) for a complete description of the Containers and Tags used.

- Activity
- Behavior
- Bogey
- Bogey (Animated Bogey)
- Bogey (Steam Bogey)
- Bridge
- Buildable
- Buildable (Passenger Station)
- Chunky-Track
- Double-Track
- DriverCharacter
- DriverCommand
- Engine (Diesel)
- Engine (Electric)
- EngineSound
- Engine (Diesel\Electric)
- Engine (Steam)
- Environment
- Fixed Track (Simple)
- Fixed Track (Junction)
- GroundTexture (Normal)
- Groundtexture (Clutter Mesh)
- HornSound (1 Part)
- HornSound (2 Part)
- HornSound (3 Part)
- Html-Asset
- Industry (Multiple Industry)
- Industry(Coal Mine)
- Interior (Diesel)
- Interior (Electric)
- Interior (Steam)
- Library
- Map
- Mesh
- Mesh-Reducing-Track
- MOCrossing
- MOJunction
- MOSignal
- MOSpeedBoard
- Pantograph
- Paintshed-Template
- Paintshed-Skin
- Product (Coal Product)
- Product (General Goods Product)
- Product (Diesel Fuel Product)
- Product (40Ft Container Product)
- Product (Lumber Product)
- Product (Passenger Product)
- Product-Category
- Profile
- Scenery
- Scenery-Trackside
- Steam-Engine
- Texture
- Texture-Group
- Track
- TrackSound
- TrainCar (Coal Hopper)
- TrainCar (Diesel Engine)
- TrainCar (Electric Engine)
- TrainCar (Rollingstock)
- TrainCar (Passenger Car)
- TrainCar (Steam Engine)
- Tunnel
- TurnTable (Animated)
- TurnTable (Not animated)
- Water2

Activity

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

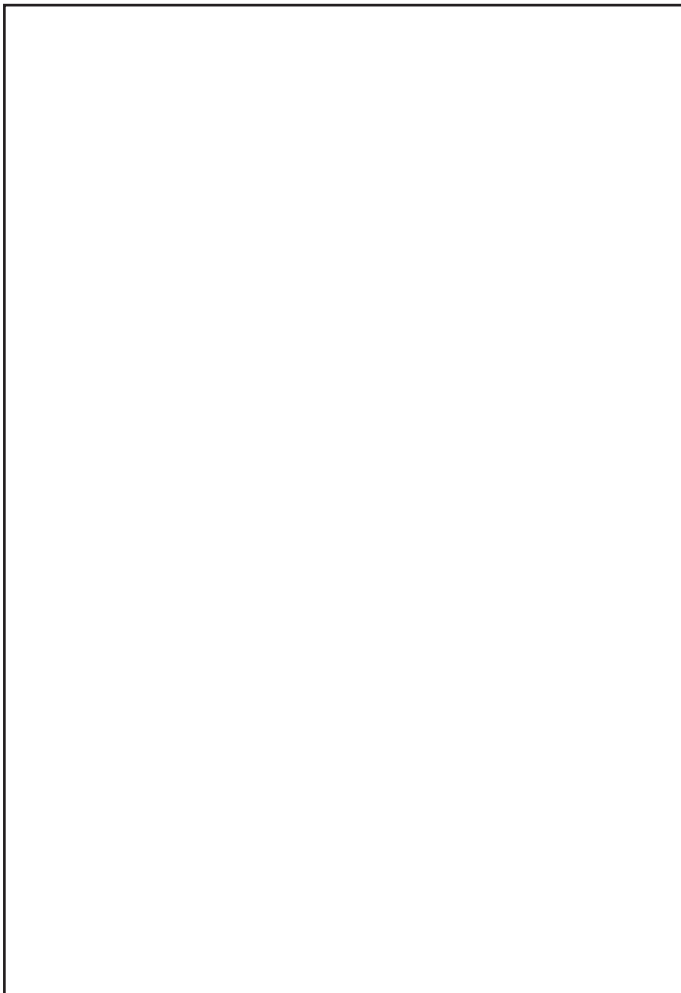
thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings



Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Behavior

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

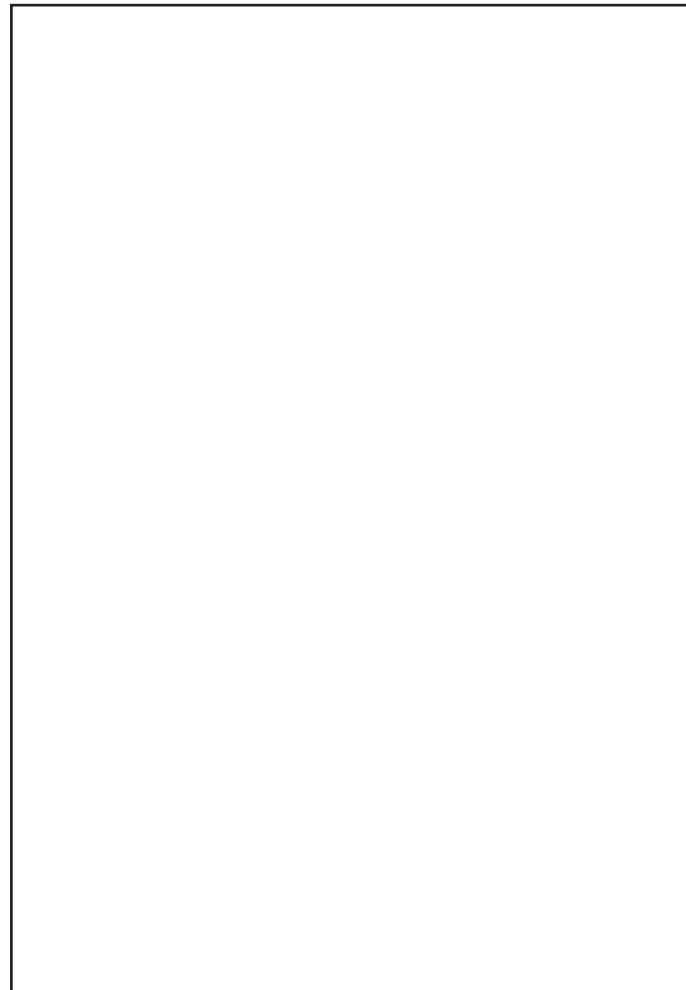
config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

Download this asset

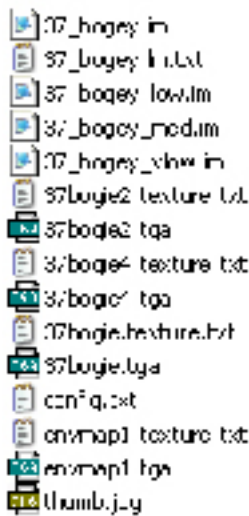
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Bogey

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

37_bogey_lm.txt - Level of Detail (or 'LOD') file. [See the section on LOD meshes ain Chapter 8 for more information.](#)

37_bogey_vlow.im - Lowest quality LOD mesh.

37_bogey_low.im - Low quality LOD mesh.

37_bogey_med.im - Medium quality LOD mesh

37_bogey.im - The bogey mesh file.

37bogie2.tga, 37bogie4.tga, 37bogie.tga, envmap1.tga - Various texture files.

37bogie2.texture.txt, 37bogie4.texture.txt, 37bogie.texture.txt, envmap1.texture.txt - Various texture.txt files. [See the section on Texture.txt files at the beginning of this chapter for more information.](#)

File Listings

config.txt	
kuid	<kuid:56113:1228>
trainz-build	2.5
category-class	"ZB"
username	"testBogey"
kind	"bogey"
direct-drive	0
category-region	"00"
category-era	"2000s;2010s"
description "Test Bogey. This bogey is based on the Class 37 Green bogey and is oriented to be used on that particular vehicle."	
mesh-table	
default	
mesh	"37_bogey.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

37_bogey_lm.txt	
version 1.0	
offset = 0.01;	
calcPoint = center;	
multiplier = 1.0;	
animationCutOff = 0.00;	
mesh("0.1") {	
name="37_bogey_vlow.im";	
}	
mesh("0.2")	
{	
name="37_bogey_low.im";	
}	
mesh("0.3")	

37_bogey.im.txt cont.

```
{  
  name="37_bogey_med.im";  
}  
mesh("1.0")  
{  
  name="37_bogey.im";  
}
```

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Bogey (Animated Bogey)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

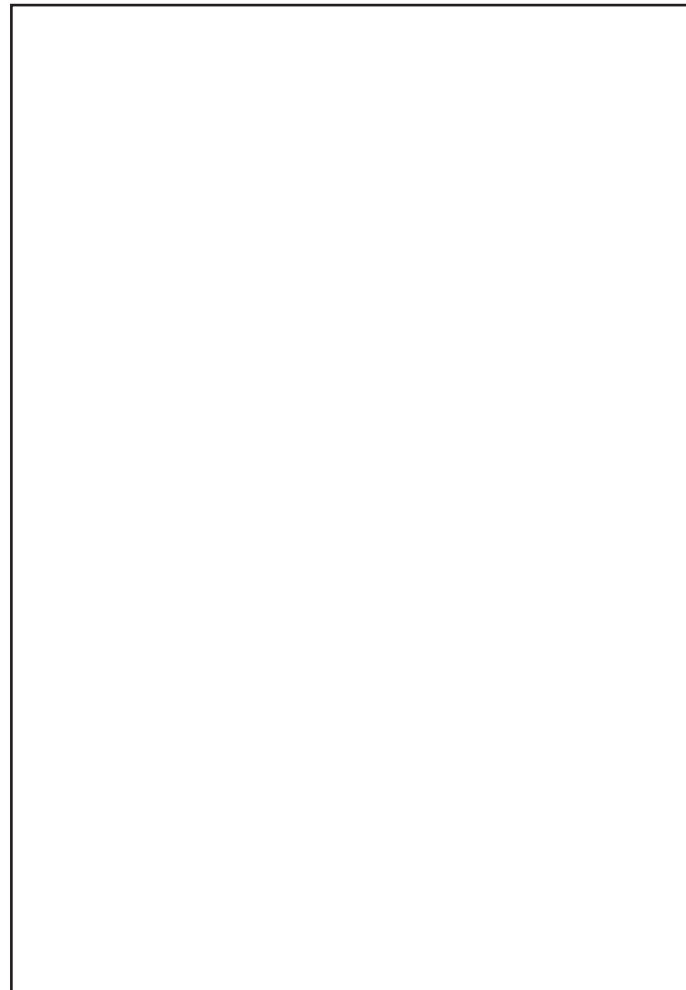
config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Bogey (Steam Bogey)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

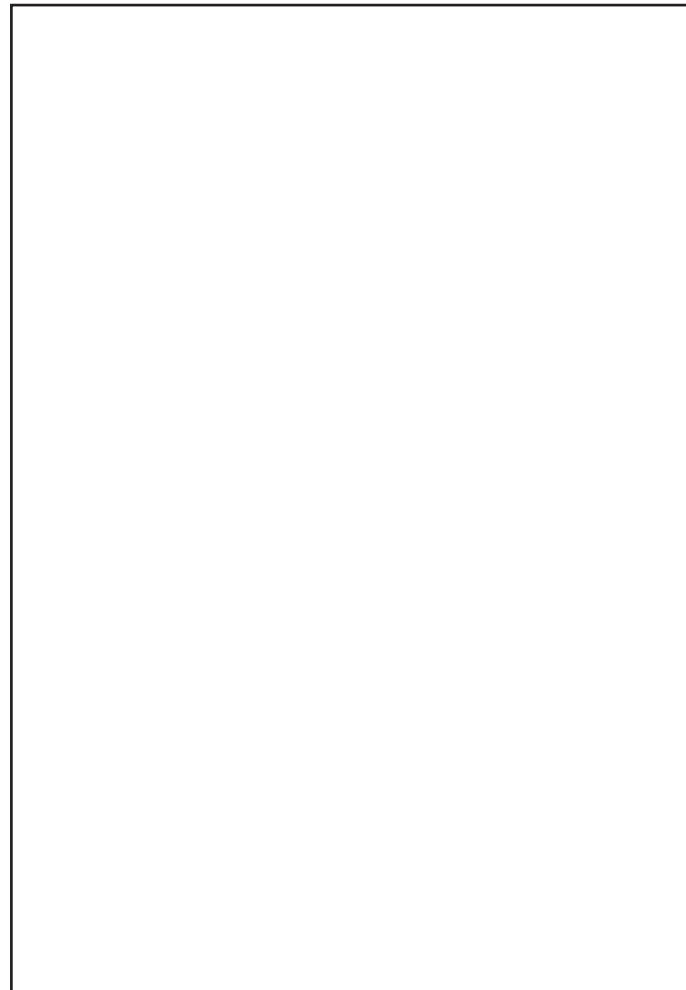
config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

Download this asset

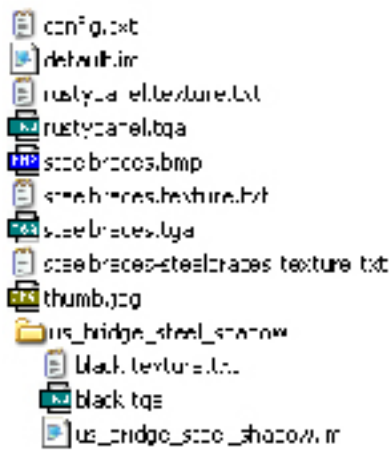
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Bridge

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

default.im - The middle section of the tunnel asset. This mesh is also used as the preview image. Must be named "default.im" and placed in the base directory.

rustypanel.texture.txt, rustypanel.tga, steelbraces.bmp, steelbraces.texture.txt, steelbraces.tga, steelbraces-steelbraces.texture.txt, black.texture.txt, black.tga - Texture files used by the indexed meshes of this bridge asset.

us_bridge_steel_shadow.im - The indexed mesh file of the bridge shadow.

File Listings

config.txt	
bendy	0
carrate	0
casts_shadows	1
endlength	0
grounded	0
isroad	0
istrack	1
length	20
repeats	1
rgb	200,100,0

shadows	0
upright	0
visible-on-minimap	1
width	7.9
kuid	<kuid:56113:1005>
trainz-build	2.5
category-class	"AA"
category-region	"AF"
category-era	"1830s"
username	"testBridge"
kind	"bridge"
bridgetrack	<kuid:-3:10049>
height	-15
trackoffsets	2.5,0
description	"Test Bridge asset."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

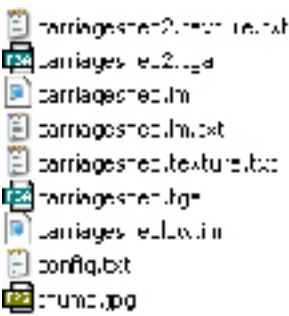
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Buildable

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- carriagedata2.texture.txt, carriagedata.texture.txt** - Various texture.txt files. See the section on LOD meshes in Chapter 8 for more information.
- carriagedata2.tga, carriagedata.tga** -Various texture files.
- carriagedata.im** -The buildable mesh file.
- carriagedata.im.txt** -Level of Detail (or ‘LOD’) file. See the section on LOD meshes at the beginning of this chapter for more information.
- carriagedata_low.im** - The low quality LOD mesh. Referenced in the carriagedata.im.txt file.

File Listings

config.txt	
kuid	<kuid:56113:1007>
trainz-build	2.5
category-class	“BB”
category-region	“AU”
category-era	“2010s”
username	“testBuildable”
kind	“buildable”
light	1
description	“This is a test Buildable asset.”
mesh-table	
default	

mesh	“carriagedata.im”
auto-create	1
attached-track	
track_0	
track	<kuid:61119:38114>
vertices	
0	“a.track0a”
1	“a.track0b”
track_1	
track	<kuid:61119:38114>
vertices	
0	“a.track0a”
1	“a.track0b”
kuid-table	
0	<kuid:61119:38114>
thumbnails	
0	
width	240
height	180
image	“thumb.jpg”

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Buildable (Passenger Station)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

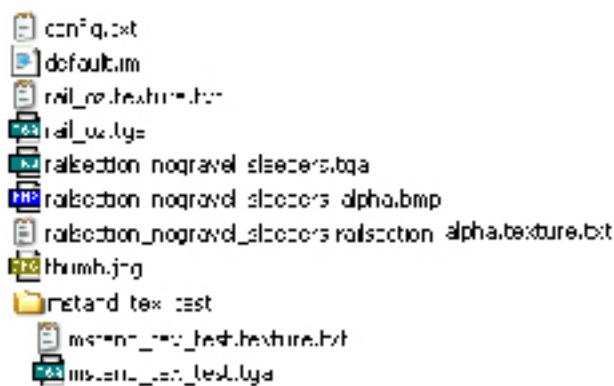
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Chunky-Track

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

default.im - The indexed mesh file used for the Chunky Track asset.

railsection_nogravel_sleepers.tga, **rail_oz.texture.txt**, **rail_oz.tga** - The texture files of the preview window asset.

railsection_nogravel_sleepers_alpha.bmp, **railsection_nogravel_sleepers-railsection_alpha.texture.txt** - The alpha texture files of the preview window asset.

mstand_tex_test.texture.txt, **mstand_tex.tga** - The texture files of the in-game Chunky Track asset.

File Listings

config.txt	
bendy	1
carrate	0
casts_shadows	0
endlength	0
grounded	0.4
isroad	0
istrack	1
length	4
repeats	1
rgb	255,200,0
shadows	0

upright	0
visible-on-minimap	1
width	4
kuid	<kuid:56113:1004>
trainz-build	2.5
category-class	"TF"
category-region	"AU"
category-era	"2000s"
username	"testChunky-Track"
kind	"track"
chunky_mesh	"mstand_tex_test"
chunky_info	0,2,1.2,0.2,0.85,0.3,0.7
description	"Test Chunky Track."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

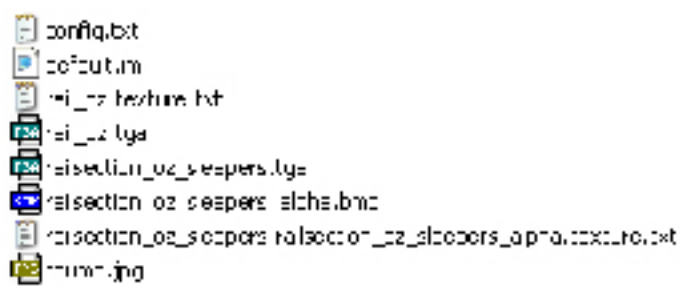
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Double-Track

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- default.im** - Though the double track references a “single track” asset to be used as the track in-game, an actual double track model must be present to be used as the preview mesh in the surveyor panel.

This indexed mesh is not referenced in the config.txt of the asset. In order to be used, the indexed mesh MUST be named “default.im”.

rail_oz.texture.txt, rail_oz.tga, railsection_oz_sleepers.tga, railsection_oz_sleepers_alpha.bmp, railsection_oz_sleepers-railsection_oz_sleepers_alpha.texture.txt - The texture files used for the “default.im” indexed mesh.

File Listings

config.txt	
bendy	1
carrate	55
casts_shadows	0
endlength	40

grounded	0.4
isroad	0
istrack	1
length	4
repeats	1
rgb	0,0,0
shadows	0
upright	0
visible-on-minimap	1
width	7.9
kuid	<kuid:56113:1001>
trainz-build	2.5
category-class	“TR”
username	“testDoubleTrack”
kind	“bridge”
bridgetrack	<kuid:-1:100396>
height	0
trackoffsets	-2.5,2.5
category-region	“00”
category-era	“1840s”
description	“Test double track asset.”
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

DriverCharacter

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

cafe64.texture.txt - The texture file which references cafe64.tga.

cafe64.tga - 64x64 Targa Image file.

cafe32.bmp - The 32x32 thumbnail image used for the small driver portraits (in the “Driver Settings” menu, or the train properties in driver).

File Listings

config.txt

kuid	<kuid:56113:1236>
trainz-build	2.5
category-class	“OHD”
category-region	“AU”
category-era	“1930s;1940s;1950s;1960s;1970s;1980s;1990s;2000s;2010s”
username	“testHenk”
kind	“drivercharacter”
face-texture	“Cafe64.texture”
mesh	<kuid:-3:10128>
description	“This is Henk.”

thumbnails

0

image	“thumb.jpg”
width	240
height	180

1

image	“cafe32.bmp”
width	32
height	32

cafe64.texture.txt

Primary=Cafe64.tga

Alpha=Cafe64.tga

Tile=none

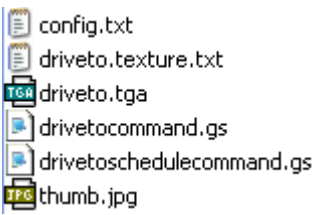
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Driver-Command

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

driveto.texture.txt - The texture.txt file. [See the section on Texture.txt files at the beginning of this chapter for more information.](#)

driveto.tga - The texture file for the drivercommand icon.

drivetocommand.gs - The trainz script file referenced in the config.txt.

drivetoschedulecommand.gs - A trainz script file which is a dependency of "drivercommand.gs".

File Listings

config.txt	
kuid	<kuid:56113:1268>
trainz-build	2.5
category-class	"YD"
category-region	"00"
category-era	"1800s;1810s;1820s;1830s;1840s"
username	"testDriverCommand"
kind	"drivercommand"
supports-null-driver-character	1
script	"DriveToCommand.gs"
class	"DriveToCommand"
description	"Test command. This does the same thing as 'Drive To'."
string-table	
description	"Allows a driver

character to take a train either to a destination industry or a specific track in a destination industry."	
driver_command_drive_to	"Drive To "
kuid-table	
command-sounds	<kuid:-3:10219>
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"driveto.tga"
width	64
height	64

DriverCommand.gs	
//	
// DriverToCommand.gs	
//	
// Copyright (C) 2003 Auran Developments Pty Ltd	
// All Rights Reserved.	
//	
include "DriverCommand.gs"	
include "World.gs"	
include "Browser.gs"	
include "KUID.gs"	
include "Industry.gs"	
include "DriveToScheduleCommand.gs"	
//	
// Driver command that allows a driver character to take a train either to a destination industry	
// or a specific track in a destination industry.	
//	
class DriveToCommand isclass DriverCommand	

```

{
//
// Initialize parent object and add handlers to process
messages for this driver command object.
//
public void Init(Asset asset)
{
    inherited(asset);
    AddHandler(me, "DriveToIndustry", null, "DriveTo");
    AddHandler(me, "DriveToIndustryTrack", null,
"DriveToTrack");
}
//
// Adds industry destination menu items for all industries to
the given menu along with submenus of
// destination tracks for all industries added.
//
public void AddCommandMenuItem(DriverCharacter driver,
Menu menu)
{
    Train train;
    if (driver)
    {
        train = driver.GetTrain();
        if (!train)
            return;
    }
    StringTable strTable = GetAsset().GetStringTable();
    Menu industriesMenu = Constructors.NewMenu();
    GameObject[] industryList = World.GetIndustryList();
    int i, industryCount = industryList.size();
    if (!industryCount)
        // we dont bother with a 'Drive To' command if there are
no industries
        return;
    for (i = 0; i < industryCount; i++)
    {

```

```

        Industry industry = cast<Industry>(industryList[i]);
        string localisedName = industry.GetLocalisedName();
        string[] locationNames = new string[0];
        string[] locationTracks = new string[0];

        industry.AppendDriverDestinations(locationNames,
locationTracks);

        if (localisedName.size())
            if (locationNames.size())
            {
                Menu submenu = Constructors.NewMenu();
                int j;
                for (j = 0; j < locationNames.size(); j++)

                    if (locationNames[j] and locationNames[j].size() and
locationTracks[j] and locationTracks[j].size())

                        submenu.AddItem(locationNames[j]
, me, "DriveToIndustryTrack", industry.GetId() + " " +
locationTracks[j]);

                else if (train)

                    train.Exception("Error in 'track names' of industry '"
+ localisedName + "'");

                industriesMenu.AddSubmenu(localisedName + ">",
submenu);
            }
            else

                industriesMenu.AddItem(localisedName, me,
"DriveToIndustry", industry.GetName());
        }
        industriesMenu.SubdivideItems();
        menu.AddSubmenu(strTable.GetString("driver_command_
drive_to") + ">", industriesMenu);
    }
//
// Called by either DriveTo() or DriveToTrack() to play one of
4 random driver acknowledgments.
//

```

```

void PlayConfirmation(void)
{
    KUID kuid = GetAsset().LookupKUIDTable("command-sounds");

    Library libCommandSounds = World.GetLibrary(kuid);
    if (libCommandSounds)
    {
        libCommandSounds.LibraryCall("PlayConfirmation", null, null);
    }
}

DriverScheduleCommand CreateScheduleCommand(DriverCharacter driver, Soup soup)
{
    DriveToScheduleCommand cmd = new DriveToScheduleCommand();

    cmd.Init(driver, me);

    cmd.SetParent(me);

    cmd.SetProperties(soup);

    return cast<DriverScheduleCommand>cmd;
}

//
// Handler method to drive a train to an industry (no specific destination track though).
//
void DriveTo(Message msg)
{
    DriverCommands commands = GetDriverCommands(msg);

    DriverCharacter driver = cast<DriverCharacter>(msg.src);

    string industryName = msg.minor;

    // schedule our command

    Soup soup = Constructors.NewSoup();

    soup.SetNamedTag("industryName", industryName);

    DriveToScheduleCommand cmd = cast<DriveToScheduleCommand>CreateScheduleCommand(driver, soup);

    commands.AddDriverScheduleCommand(cmd);

    if (driver)

```

```

    PlayConfirmation();
}

//
// Handler method to drive a train to a specific track in an industry.
//
void DriveToTrack(Message msg)
{
    DriverCommands commands = GetDriverCommands(msg);

    DriverCharacter driver = cast<DriverCharacter>(msg.src);

    string msgData = msg.minor;

    int industryId = Str.UnpackInt(msgData);

    string trackName = Str.UnpackString(msgData);

    GameObject industry = Router.GetGameObject(industryId);

    if (!industry)

        return;

    // schedule our command

    Soup soup = Constructors.NewSoup();

    soup.SetNamedTag("industryName", industry.GetName());

    soup.SetNamedTag("trackName", trackName);

    DriveToScheduleCommand cmd = cast<DriveToScheduleCommand>CreateScheduleCommand(driver, soup);

    commands.AddDriverScheduleCommand(cmd);

    if (driver)

        PlayConfirmation();
}
};

```

DriveToScheduleCommand.gs

```
//  
// DriveToScheduleCommand.gs  
//  
// Copyright (C) 2003 Auran Developments Pty Ltd  
// All Rights Reserved.  
//  
include "DriverCommand.gs"  
include "World.gs"  
include "Browser.gs"  
include "KUID.gs"  
include "Industry.gs"  
include "DriveToCommand.gs"  
include "Schedule.gs"  
//  
// Driver schedule command used by DriveToCommand to get  
// a driver character to take a train to  
// a specific track on an industry.  
//  
class DriveToScheduleCommand isclass  
DriverScheduleCommand  
{  
    public string industryName; // Name of the industry to drive  
    to.  
    public string trackName; // Name of the track in the  
    industry to drive to.  
    DriveToCommand parent;  
    public void SetParent(DriveToCommand newParent)  
    {  
        parent = newParent;  
    }  
    //  
    // Starts executing this schedule command on the given driver  
    character.  
    //  
    public bool BeginExecute(DriverCharacter driver)  
    {
```

```
        Train train = driver.GetTrain();  
        if (!train)  
        {  
            // cant drive if we dont have a train  
            return false;  
        }  
        Industry industry = cast<Industry>(Router.GetGameObject(i  
ndustryName));  
        if (!industry)  
        {  
            // cant drive to an industry which doesn't exist  
            return false;  
        }  
        return driver.NavigateToIndustry(industry, trackName);  
    }  
    // we should really implement EndExecute() to allow the  
    game to determine the success of this command  
    //  
    // Provides an icon for this command so it can be seen on the  
    driver's schedule. Uses the industry  
    // icon to indicate the destination.  
    //  
    public object GetIcon(void)  
    {  
        Industry industry = cast<Industry>(Router.GetGameObject(i  
ndustryName));  
        return cast<object>industry;  
    }  
    public string GetTooltip(void)  
    {  
        StringTable strTable = GetAsset().GetStringTable();  
        string userTrackName = trackName;  
        Industry industry = cast<Industry>(Router.GetGameObject(i  
ndustryName));  
        if (industry)  
        {  
            string[] destNames = new string[0];  
            string[] destTracks = new string[0];  
            industry.AppendDriverDestinations(destNames,  
destTracks);  
            int i;
```

```

for (i = 0; i < destNames.size(); i++)
{
    if (destTracks[i] == trackName)
    {
        userTrackName = destNames[i];
        break;
    }
}

return strTable.GetString("driver_command_drive_to") +
industryName + " (" + userTrackName + ")";
}

public Soup GetProperties(void)
{
    Soup soup = Constructors.NewSoup();
    // Save support
    // Save the properties to the soup, then return the soup
    soup.SetNamedTag("industryName", industryName);
    soup.SetNamedTag("trackName", trackName);
    return soup;
}

public void SetProperties(Soup soup)
{
    // Load support
    // Setup the properties from the passed in soup
    industryName = soup.GetNamedTag("industryName");
    trackName = soup.GetNamedTag("trackName");
}
};

```

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Engine (Diesel)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

config.txt	
kuid	<kuid:56113:1230>
trainz-build	2.5
category-class	"ZE"
category-region	"ES"
category-era	"2000s;2010s"
username (Diesel)"	"testEngine
kind	"engine"
description	"Test EngineSpec."
flowsize	
trainbrakepipe	170000
epreservoirpipe	0.1
no3pipe	0.1
no4pipe	0.1
auxreservoirvent	0.1
auxreservoir_no3	0.1
auxreservoir_trainbrakepipe	0.1
autobrakecylindervent	0.1
auxreservoir_autobrakecylinder	0.1
equaliser_mainreservoir	0.06
equaliservent	0.06
equaliserventhandleoff	0.1
equaliserventemergency	0.1
no3pipevent	1.5

config.txt cont.	
no3pipe_mainreservoir	0.1
compressor	5
trainbrakepipe_reservoir	1
trainbrakepipevent	0.06
no3pipe_autobrakecylinder	0.1
epreservoirpipe_autobrakecylinder	0.1
mainreservoir_ep	0.1
vacuumbrakepipe	0.1
vacuumbrakepipereleasevent	0.1
vacuumbrakepipevent	0.1
vacuumbrakereservoir_vacuumbrakepipe	0.1
vacuumbrakecylinder_vacuumbrakepipe	0.1
highspeedexhauster_vacuumbrakepipe	0.1
volume	
scale	1
trainbrakepipe	0.2
epreservoirpipe	0.2
no3pipe	0.2
no4pipe	0.2
auxreservoir	0.0384678
autobrakecylinder	0.00969387
vacuumbrakepipe	0
vacuumbrakereservoir	0
vacuumbrakecylinder	0
mainreservoir	2
equaliser	0.5
independantbrakecylinder	0.0103239
pressure	
scale	1
compressor	0.011248
mainreservoir	0.0081548
highspeedexhauster	0
brakepipe	0.00736041
brakeinitial	0.00665741

config.txt cont.		config.txt cont.	
brakefull	0.00553261	0	"30"
indbrakefull	0.00553261	5	"25"
trainbrakepipe_start	0.00553261	10	"15"
epreservoirpipe_start	0	12	"0"
no3pipe_start	0	2	
no4pipe_start	0	0	"55"
auxreservoir_start	0.00553261	5	"48"
autobrakecylinder_start	0.00560291	10	"40"
vacuumbrakepipe_start	0	15	"30"
vacuumbrakereservoir_start	0	30	"0"
vacuumbrakecylinder_start	0	3	
mainreservoir_start	0.00946941	0	"90"
equaliser_start	0.00553261	5	"60"
independantbrakecylinder_start	0.00560291	10	"45"
mass		15	"40"
scale	1	30	"0"
fuel	6.2156e+006	4	
motor		2	"120"
resistance	1	5	"70"
adhesion	5	10	"60"
maxvoltage	600	15	"55"
maxspeed	33.33	30	"0"
brakeratio	55000	5	
max-accel	3000	0	"180"
max-decel	8500	5	"140"
throttle-notches	32	10	"70"
axle-count	4	15	"55"
surface-area	150	30	"0"
moving-friction-coefficient	0.03	6	
air-drag-coefficient	0.0025	0	"220"
throttle-power		5	"170"
0		10	"110"
0	"0"	15	"80"
1		30	"0"

7		22	"0"
0	"230"	4	
5	"220"	1.333	"0"
10	"200"	4	"80"
15	"100"	10	"60"
30	"0"	20	"20"
8		25	"0"
0	"250"	5	
3.5	"200"	1.333	"0"
5	"160"	5	"90"
10	"130"	10	"70"
15	"100"	25	"25"
30	"90"	29	"0"
32	"60"	6	
34	"0"	1.333	"0"
dynamic-brake		5	"150"
0		10	"80"
0	"0"	29	"70"
1		32	"0"
1.333	"0"	7	
2	"30"	1.333	"0"
5	"25"	5	"150"
10	"15"	10	"100"
12	"0"	32	"60"
2		36	"0"
1.333	"0"	8	
3	"50"	1.33	"0"
10	"35"	5	"150"
14	"20"	10	"100"
15	"0"	36	"50"
3		40	"0"
1.333	"0"	thumbnails	
3	"60"	0	
10	"40"	image	"thumb.jpg"
17	"20"	width	240

height

180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Engine (Electric)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

config.txt.

kuid	<kuid:56113:1234>
trainz-build	2.5
category-class	"ZE"
category-region	"CZ"
category-era	"1980s;1990s;2000s;2010s"
username	"testEngine (Electric)"
kind	"engine"
description	"Test electric engine asset. Based on the TGV."
flowsize	
trainbrakepipe	170000
epreservoirpipe	0.1
no3pipe	0.1
no4pipe	0.1
auxreservoirvent	0.1
auxreservoir_no3	0.1

auxreservoir_trainbrakepipe	0.1
autobrakecylindervent	0.1
auxreservoir_autobrakecylinder	0.1
equaliser_mainreservoir	0.06
equaliservent	0.06
equaliserventhandleoff	0.1
equaliserventemergency	0.1
no3pipeevent	1.5
no3pipe_mainreservoir	0.1
compressor	10
trainbrakepipe_reservoir	1
trainbrakepipeevent	0.06
no3pipe_autobrakecylinder	0.1
epreservoirpipe_autobrakecylinder	0.1
mainreservoir_ep	0.1
vacuumbrakepipe	0.1
vacuumbrakepipereleasevent	0.1
vacuumbrakepipevent	0.1
vacuumbrakereservoir_vacuumbrakepipe	0.1
vacuumbrakecylinder_vacuumbrakepipe	0.1
highspeedexhauster_vacuumbrakepipe	0.1
volume	
scale	1
trainbrakepipe	0.2
epreservoirpipe	0.2
no3pipe	0.2
no4pipe	0.2
auxreservoir	0.0384678
autobrakecylinder	0.00969387
vacuumbrakepipe	0
vacuumbrakereservoir	0
vacuumbrakecylinder	0
mainreservoir	0.9
equaliser	0.5
independantbrakecylinder	0.0103239

pressure		moving-friction-coefficient	0.03
scale	1	air-drag-coefficient	0.00017
compressor	0.011248	throttle-power	
mainreservoir	0.0081548	0	
highspeedexhauster	0	0	“0”
brakepipe	0.00609501	1	
brakeinitial	0.00560291	0	“20”
brakefull	0.00447811	1.5	“16”
indbrakefull	0.00447811	3	“10”
trainbrakepipe_start	0.00447811	10	“1”
epreservoirpipe_start	0	2	
no3pipe_start	0	0	“40”
no4pipe_start	0	1.5	“28”
auxreservoir_start	0.00447811	3	“20”
autobrakecylinder_start	0.00507566	5	“10”
vacuumbrakepipe_start	0	10	“1”
vacuumbrakereservoir_start	0	3	
vacuumbrakecylinder_start	0	0	“52”
mainreservoir_start	0.00876641	1.5	“36”
equaliser_start	0.00447811	3	“30”
independantbrakecylinder_start	0.00507566	10	“20”
mass		18	“10”
scale	1	25	“1”
fuel	“6.2156e+006”	4	
motor		0	“92”
resistance	1	1.5	“76”
adhesion	3	3	“58”
maxvoltage	600	5	“50”
maxspeed	90	15	“40”
brakeratio	50000	30	“24”
max-accel	12500	40	“1”
max-decel	175000	5	
throttle-notches	32	0	“132”
axle-count	4	1.5	“116”
surface-area	80	3	“98”

15	"92"	10	"30"
40	"60"	30	"0"
50	"30"	2	
60	"1"	0	"150"
6		5	"120"
0	"172"	10	"110"
1.5	"156"	25	"90"
3	"136"	35	"50"
50	"106"	3	
60	"52"	0	"200"
70	"36"	5	"180"
80	"1"	10	"150"
7		25	"110"
0	"160"	50	"60"
1.5	"150"	4	
30	"140"	0	"270"
50	"130"	3.5	"230"
70	"70"	5	"190"
80	"50"	10	"150"
90	"1"	40	"110"
8		70	"60"
0	"160"	5	
1.5	"140"	0	"350"
5	"134"	3.5	"320"
8	"120"	5	"270"
50	"114"	10	"190"
60	"110"	40	"150"
70	"100"	70	"80"
90	"0"	6	
dynamic-brake		0	"400"
0		10	"360"
0	"0"	15	"300"
1		30	"220"
0	"100"	60	"100"
5	"75"	90	"80"

7		
0		"500"
10		"470"
15		"420"
30		"360"
60		"250"
90		"130"
105		"80"
8		
0		"600"
10		"550"
15		"460"
30		"350"
60		"220"
90		"100"
105		"90"
135		"20"
thumbnails		
0		
image		"thumb.jpg"
width		240
height		180

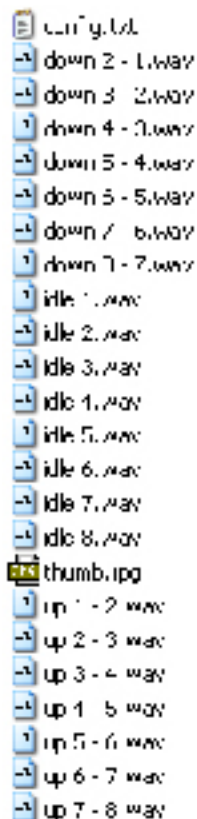
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Enginesound (Diesel\Electric)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

down 2 - 1.wav, down 3 - 2.wav, down 4 - 3.wav, down 5 - 4.wav, down 6 - 5.wav, down 7 - 6.wav, down 8 - 7.wav - The various sound files used for “slowing down” sounds relative to each of the 8 notches.

idle 1.wav, idle 2.wav, idle 3.wav, idle 4.wav, idle 5.wav, idle 6.wav, idle 7.wav, idle 8.wav - The various sound files used for “idle” sounds relative to each of the 8 notches.

up 1 - 2.wav, up 2 - 3.wav, up 3 - 4.wav, up 4 - 5.wav, up 5 - 6.wav, up 6 - 7.wav, up 7 - 8.wav - The various sound files used for “speeding up” sounds relative to each of the 8 notches.

File Listings

config.txt	
kuid	<kuid:56113:1242>
trainz-build	2.5
category-class	“ZS”
category-region	“UK”
category-era	“1960s;1970s;1980s;1990s;2000s”
username	“testEngineSound (Diesel or Electric)”
kind	“enginesound”
description	“Test Enginesound. Based on the BR 37 Sounds.”
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

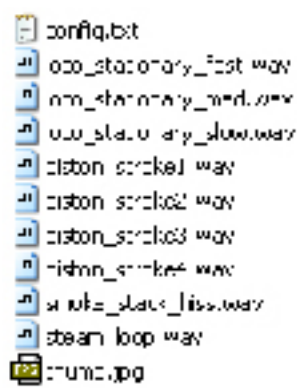
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Enginesound (Steam)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

loco_stationary_fast.wav, loco_stationary_med.wav, loco_stationary_slow.wav - These file are the steam engine idling sounds played after the steam engine is stationary for 1, 2 and 3 minutes.

piston_stroke1.wav, piston_stroke2.wav, piston_stroke3.wav, piston_stroke4.wav - Piston stoke sounds, played every 180 degrees revolution of the piston wheel played in sequence and repeated up to about 40 kph.

smoke_stack_hiss.wav - The general hiss from the smoke stack.

steam_loop.wav - From 40 kph upwards, the following sound loop is cross-faded as the piston sounds die off. The loop is pitched shifted (through code) relative to the locomotive's velocity.

File Listings

config.txt	
kuid	<kuid:56113:1243>
trainz-build	2.5
category-class	"ZS"
category-region	"US"
category-era	"1960s;1970s;1980s;1990s;2000s"
username	"testEngineSound (Steam)"
kind	"enginesound"
description	"Test Steam Engine Sounds. Based on the PB15."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

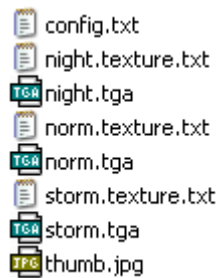
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Environment

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

night.texture.txt, night.tga - The image files used for the environment when it's night time.

norm.texture.txt, norm.tga - The image files used for the environment when it's day time.

storm.texture.txt, storm.tga - The image files used for the environment when the conditions are stormy.

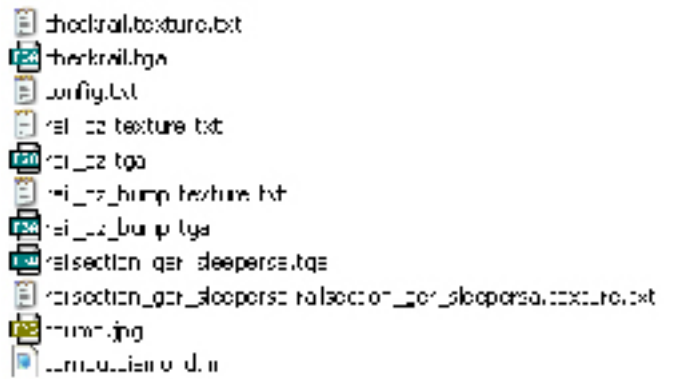
File Listings

config.txt	
kuid	<kuid:56113:1227>
trainz-build	2.5
category-class	"ES"
category-region	"00"
category-era	"2010s"
username	"testEnvironment"
kind	"environment"
normal	"norm"
storm	"storm"
night	"night"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Fixed Track (Simple)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

checkrail.texture.txt, checkrail.tga, rail_oz.texture.txt, rail_oz.tga, rail_oz_bump.texture.txt, rail_oz_bump.tga, railsection_ger_sleepersa.tga, railsection_ger_sleepersa_railsection_ger_sleepersa.texture.txt - The texture files used in the indexed mesh of this asset.

turnoutdiamond.im - The indexed mesh for this Fixed Track asset.

File Listings

config.txt	
kuid	<kuid:56113:1017>
trainz-build	2.5
category-class	"TF"
category-region	"00"
category-era	"1980s;1990s;2000s;2010s"
username	"testFixedTrack(Diamond)"
kind	"fixedtrack"
description	"Test Fixed Track asset."
height-range	-50,50
preview-mesh-kuid	<kuid:171456:60520>
use-gradient-track	1
mesh-table	
default	
mesh	"turnoutdiamond.im"

auto-create	1
effects	
arrow0	
kind	"attachment"
att	"a.track0a"
default-mesh	<kuid:-3:10092>
surveyor-only	1
arrow1	
kind	"attachment"
att	"a.track0b"
default-mesh	<kuid:-3:10092>
surveyor-only	1
arrow2	
kind	"attachment"
att	"a.track1a"
default-mesh	<kuid:-3:10092>
surveyor-only	1
arrow3	
kind	"attachment"
att	"a.track1b"
default-mesh	<kuid:-3:10092>
surveyor-only	1
attached-track	
track0	
track	<kuid:67598:38001>
useadjoiningtracktype	0
vertices	
0	"a.track0a"
1	"a.track0b"
track1	
track	<kuid:67598:38001>
useadjoiningtracktype	0
vertices	
0	"a.track1a"
1	"a.track1b"

kuid-table	
0	<kuid:-3:10092>
1	<kuid:67598:38001>
2	<kuid:171456:60520>
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

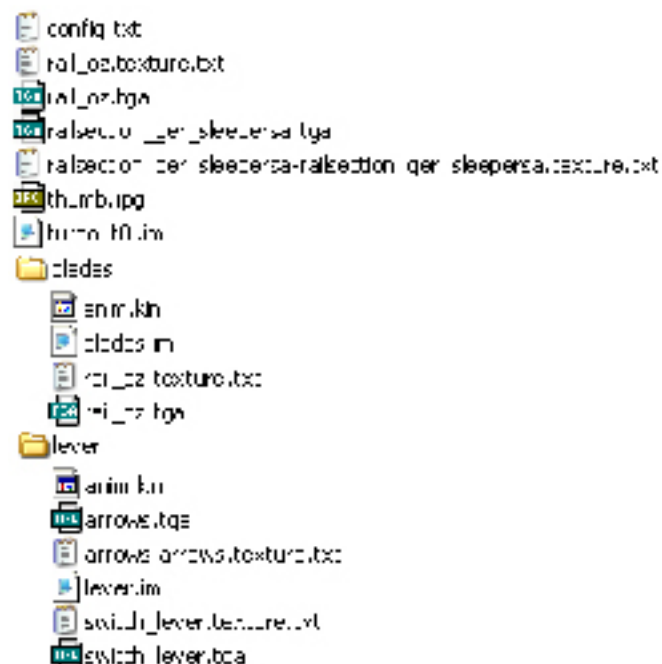
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Fixed Track (Junction)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

rail_oz.texture.txt, rail-oz.tga, railsection_ger_sleepersa.tga, railsection_ger_sleepersa-railsection_ger_sleepersa.texture.txt, arrows.tga, arrows-arrows.texture.txt, switch_lever.texture.txt, switch_lever.tga - The texture files used in the indexed meshes of this asset.

turnout8l, blades.im, lever.im - The indexed meshes used to build this asset.

anim.kin - The animation files used for both the moving track blades, and the animating lever.

File Listings

config.txt	
kuid	<kuid:56113:1018>
trainz-build	2.5
category-class	"TF"
category-region	"00"
category-era	"1950s;1970s;1980s;1990s;2000s;2010s"
username	"testFixedTrack (Junction)"
kind	"fixedtrack"
description (Junction)."	"Test fixed track asset
height-range	-50,50
preview-mesh-kuid	<kuid:171456:60560>
use-gradient-track	1
mesh-table	
default	
mesh	"turnout8l.im"
auto-create	1
effects	
arrow0	
kind	"attachment"
att	"a.track0a"
default-mesh	<kuid:-3:10092>
surveyor-only	1
arrow1	
kind	"attachment"
att	"a.track0e"
default-mesh	<kuid:-3:10092>
surveyor-only	1
arrow2	
kind	"attachment"
att	"a.track1b"
default-mesh	<kuid:-3:10092>
surveyor-only	1
blades	
mesh	"Blades/blades.im"

anim	"Blades/anim.kin"
auto-create	1
lever1	
mesh	"Lever/lever.im"
anim	"Lever/anim.kin"
auto-create	1
att-parent	"default"
att	"a.lever1"
attached-track	
track0	
track	<kuid:67598:38001>
useadjoiningtracktype	0
vertices	
0	"a.track0a"
1	"a.track0b"
2	"a.track0c"
3	"a.track0d"
4	"a.track0e"
track1	
track	<kuid:67598:38001>
useadjoiningtracktype	0
vertices	
0	"a.track0b"
1	"a.track1a"
2	"a.track1b"
junction-vertices	
0	
junction-lever-mesh	"lever0"
junction-vertex	"a.track0b"
1	
junction-lever-mesh	"lever1"
junction-vertex	"a.track0b"
2	
junction-lever-mesh	"blades"
junction-vertex	"a.track0b"

kuid-table	
0	<kuid:-3:10092>
1	<kuid:67598:38001>
2	<kuid:171456:60563>
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Groundtexture (Normal)

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- ugly.bmp** - The 128x128 bitmap image used as a ground texture.

File Listings

config.txt	
kuid	<kuid:56113:1246>
trainz-build	2.5
category-class	“GL”
category-region	“AU”
category-era	“1980s;1990s;2000s;2010s”
username	“testGroundTexture”
kind	“groundtexture”
texture	“ugly.bmp”
description example.”	“A very ugly ground texture
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

Groundtexture (Clutter Mesh)

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- ugly.bmp** - The 128x128 bitmap image used as a ground texture.

File Listings

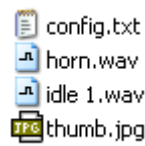
config.txt	
kuid	<kuid:56113:1247>
trainz-build	2.5
category-class	"GL"
category-region	"AF"
category-era	"1980s;1990s;2000s"
username	"testGroundTexture (Clutter-Mesh)"
kind	"groundtexture"
texture	"ugly.bmp"
clutter-mesh	<kuid:-3:10128>
description	"A very ugly Ground Texture with a whole bunch of Alastair's running around as the clutter mesh."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Hornsound (1 Part)

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

horn.wav - Railyard hornsound (non looping).

idle1.wav - Generally used for the bell sound (bell keystroke = b).

File Listings

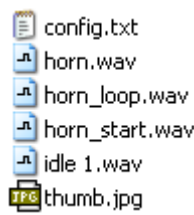
config.txt	
kuid	<kuid:56113:1269>
trainz-build	2.5
category-class	“ZH”
category-region	“AN”
category-era	“1990s;2000s;2010s”
username	“testHornsound (1 Part)”
kind	“hornsound”
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

Hornsound (2 Part)

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

horn.wav - ‘Railyard’ hornsound (non-looping).

horn_loop.wav - The looping hornsound used in ‘Driver’.

horn_start.wav - The starting sound played before the looping hornsound above.

idle 1.wav - Generally used for the bell sound (bell keystroke = b).

File Listings

config.txt	
kuid	<kuid:56113:1273>
trainz-build	2.5
category-class	“ZH”
category-region	“AN”
category-era	“1840s;1860s;1870s;1880s”
username	“testHornsound (2 Part)”
kind	“hornsound”
two-part	1
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

Hornsound (3 Part)

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- idle 1.wav** - The looping sound played while the horn is held down.
- start.wav** - The sound played when the horn is first sounded.
- stop.wav** - The sound played when the horn is released.

File Listings

config.txt	
kind	"hornsound"
kuid	<kuid:56113:1013>
trainz-build	2.5
category-class	"ZH"
category-region	"00"
category-era	"1800s"
username	"testHornsound (3 Part)"
three-part	1
description	"A 3 part horn sound."
thumbnails	
0	
image	"thumb.JPG"
width	240
height	180

HTML-Asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

index.html - The HTML page to be used in-game.

traffic.jpg - An image file used in the index.html HTML page.

File Listings

config.txt	
kuid	<kuid:56113:1003>
trainz-build	2.5
category-class	"YH"
category-region	"AG"
category-era	"1850s"
username	"testHTMLAsset"
kind	"html-asset"
description	"Test HTML Asset."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

```
index.html

<html>

<body>

    <img src = "traffic.jpg">

</body>

</html>
```

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Industry (Multiple Industry)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

concrete.texture.txt, concrete.tga, recess.texture.txt, recess.tga - The texture files used for the industry asset.

icon_multiple.tga - The image files used as an icon for the industry.

multiple_industry.im - The indexed mesh used for the Industry asset.

multiple_industry.gs - The script file used to outline the behavior of the industry.

File Listings

config.txt	
kuid	<kuid:56113:1001>
trainz-build	2.5
category-class	"BIN"
category-region	"00"
category-era	"1810s"
username	"testIndustry"
kind	"industry"
light	1
nightmode	"lamp"
script	"multipleindustry.gs"
class	"MultipleIndustry"
preview-mesh-kuid	<kuid:-3:10154>
icon0	<kuid:-3:10164>

description "Test Industry asset. Based on the Multiple Industry New."	
kuid-table	
coal	<kuid:44179:60013>
diesel	<kuid:-3:10011>
cont20ft	<kuid:-3:10014>
cont40ft	<kuid:-3:10041>
gengoods	<kuid:-3:10013>
logs	<kuid:-3:10001>
lumber	<kuid:-3:10003>
woodchips	<kuid:-3:10002>
oil	<kuid:-3:10010>
petrol	<kuid:-3:10012>
water	<kuid:-3:10004>
avgas	<kuid:-3:10045>
mesh-table	
default	
mesh	"Multiple_Industry.im"
auto-create	1
effects	
arrow0	
att	"a.track0a"
default-mesh	<kuid:-3:10092>
surveyor-only	1
kind	"attachment"
arrow1	
att	"a.track0f"
default-mesh	<kuid:-3:10092>
surveyor-only	1
kind	"attachment"
attached-track	
out_track0	
track	<kuid:-1:15>
vertices	
0	"a.track0a"

1	"a.track0b"	allowed-products	
2	"a.track0c"	0	<kuid:-3:10013>
3	"a.track0d"	logs_q	
4	"a.track0e"	size	90
5	"a.track0f"	initial-count	45
attached-trigger		product-kuid	<kuid:-3:10001>
trig0		allowed-products	
att	"a.track0b"	0	<kuid:-3:10001>
radius	10	lumber_q	
trig1		size	51
att	"a.track0c"	initial-count	25
radius	10	product-kuid	<kuid:-3:10003>
trig2		allowed-products	
att	"a.track0d"	0	<kuid:-3:10003>
radius	10	coal_q	
trig3		size	1086000
att	"a.track0e"	initial-count	543000
radius	10	product-kuid	<kuid:44179:60013>
queues		allowed-products	
20ft_cont_q		0	<kuid:44179:60013>
size	100	woodchips_q	
initial-count	50	size	726000
product-kuid	<kuid:-3:10014>	initial-count	302500
allowed-products		product-kuid	<kuid:-3:10002>
0	<kuid:-3:10014>	allowed-products	
40ft_cont_q		0	<kuid:-3:10002>
size	100	crude_oil_q	
initial-count	50	size	1940000
product-kuid	<kuid:-3:10041>	initial-count	776000
allowed-products		product-kuid	<kuid:-3:10010>
0	<kuid:-3:10041>	allowed-products	
gen_goods_q		0	<kuid:-3:10010>
size	100	diesel_q	
initial-count	50	size	1164000
product-kuid	<kuid:-3:10013>	initial-count	582000

product-kuid	<kuid:-3:10011>	queue	"gen_goods_q"
allowed-products		3	
0	<kuid:-3:10011>	amount	1
petrol_q		queue	"logs_q"
size	388000	4	
initial-count	194000	amount	1
product-kuid	<kuid:-3:10012>	queue	"lumber_q"
allowed-products		5	
0	<kuid:-3:10012>	amount	100000
water_q		queue	"coal_q"
size	15000	6	
initial-count	10000	amount	100000
animated-mesh	"default"	queue	"woodchips_q"
product-kuid	<kuid:-3:10004>	7	
allowed-products		amount	100000
0	<kuid:-3:10004>	queue	"crude_oil_q"
av_in_q		8	
size	582000	amount	100000
initial-count	271600	queue	"diesel_q"
product-kuid	<kuid:-3:10045>	9	
allowed-products		amount	100000
0	<kuid:-3:10045>	queue	"petrol_q"
processes		10	
multi_consumer_producer		amount	100
start-enabled	1	queue	"water_q"
duration	30	11	
inputs		amount	100000
0		queue	"av_in_q"
amount	1	outputs	
queue	"20ft_cont_q"	0	
1		amount	1
amount	1	queue	"20ft_cont_q"
queue	"40ft_cont_q"	1	
2		amount	1
amount	1	queue	"40ft_cont_q"

2		
amount	1	
queue	“gen_goods_q”	
3		
amount	1	
queue	“logs_q”	
4		
amount	1	
queue	“lumber_q”	
5		
amount	100000	
queue	“coal_q”	
6		
amount	100000	
queue	“woodchips_q”	
7		
amount	100000	
queue	“crude_oil_q”	
8		
amount	100000	
queue	“diesel_q”	
9		
amount	100000	
queue	“petrol_q”	
10		
amount	100	
queue	“water_q”	
11		
amount	100000	
queue	“av_in_q”	
string-table		
multi_pickupdropoff	“Multiple Pickup/Drop off”	
thumbnails		
0		
image	“icon_multiple.tga”	

width	64
height	64
1	
image	“thumb.jpg”
width	240
height	180

multipleindustry.gs
include “BaseIndustry.gs”
//
// MultipleIndustry industry
//
class MultipleIndustry isclass BaseIndustry
{
ProductQueue crudeOilQueue, dieselQueue, petrolQueue,
coalQueue, cont20ftQueue, cont40ftQueue, gengoodsQueue,
logsQueue, lumberQueue, woodchipsQueue, waterQueue,
avgasQueue;
Asset crudeOilAsset, dieselAsset, petrolAsset, coalAsset,
cont20ftAsset, cont40ftAsset, gengoodsAsset, logsAsset,
lumberAsset, woodchipsAsset, waterAsset, avgasAsset;
bool animating = false;
bool processing = false;
bool scriptletEnabled = true;
// Track if they only supply some of the logs that were
requested in the waybill.
int avWBRemain = 0;
int cont20WBRemain = 0;
int cont40WBRemain = 0;
int goodsWBRemain = 0;
int logWBRemain = 0;
int lumberWBRemain = 0;
int coalWBRemain = 0;
int woodchipWBRemain = 0;
int oilWBRemain = 0;
int dieselWBRemain = 0;
int petrolWBRemain = 0;

```

int waterWBRemain = 0;

//

//

//

bool TriggerSupportsStoppedLoad(Vehicle vehicle, string
triggerName)
{
    bool vehicleToTrain = vehicle.GetFacingRelativeToTrain();

    int direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (!vehicleToTrain)

        direction = -direction;

    // Are we up to the furthest trigger away from the side we
entered for diesel?

    if (direction == Vehicle.DIRECTION_BACKWARD and
triggerName == "trig3")

        return true;

    if (direction == Vehicle.DIRECTION_FORWARD and
triggerName == "trig0")

        return true;

    // If the train has already stopped, then fall thru and allow
this load as well

    if (triggerName == "trig0" or triggerName == "trig1" or
triggerName == "trig2" or triggerName == "trig3")

    {
        if (vehicle.GetMyTrain().IsStopped())

            return true;

    }

    return false;
}

void PerformStoppedLoad(Vehicle vehicle, string
triggerName)
{
    if (triggerName == "trig0" or triggerName == "trig1" or
triggerName == "trig2" or triggerName == "trig3")

    {

        bool avWBModified = false;

        bool cont20WBModified = false;

        bool cont40WBModified = false;

```

```

bool goodsWBModified = false;

bool logWBModified = false;

bool lumberWBModified = false;

bool coalWBModified = false;

bool woodchipWBModified = false;

bool oilWBModified = false;

bool dieselWBModified = false;

bool petrolWBModified = false;

bool waterWBModified = false;

int spaceAvailable;

LoadingReport report;

int direction;

    if (itc.IsTrainCommand(vehicle.GetMyTrain(), Industry.
LOAD_COMMAND))

    {

        // Attempt to load everything! ;)

        //

        // Load the crudeoil

        //

        if (GetProcessOutput("multi_consumer_producer",
crudeOilQueue, crudeOilAsset) > 0)

        {

            spaceAvailable = crudeOilQueue.GetQueueCount();

            report = CreateLoadingReport(crudeOilQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.LoadProduct(report);

            // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

            if (report.amount > 0)

                oilWBModified = true;

```

```

}

//

// Load the diesel

//

if (GetProcessOutput("multi_consumer_producer",
dieselQueue, dieselAsset) > 0)

{

    spaceAvailable = dieselQueue.GetQueueCount();

    report = CreateLoadingReport(dieselQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        dieselWBModified = true;

}

//

// Load the petrol

//

if (GetProcessOutput("multi_consumer_producer",
petrolQueue, petrolAsset) > 0)

{

    spaceAvailable = petrolQueue.GetQueueCount();

    report = CreateLoadingReport(petrolQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

```

```

vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        petrolWBModified = true;

}

//

// Load the coal

//

if (GetProcessOutput("multi_consumer_producer",
coalQueue, coalAsset) > 0)

{

    spaceAvailable = coalQueue.GetQueueCount();

    report = CreateLoadingReport(coalQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        coalWBModified = true;

}

//

// Load the cont20

//

if (GetProcessOutput("multi_consumer_producer",
cont20ftQueue, cont20ftAsset) > 0)

{

    spaceAvailable = cont20ftQueue.GetQueueCount();

    report = CreateLoadingReport(cont20ftQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

```

```

        if (direction == Vehicle.DIRECTION_FORWARD)

            report.sideFlags = LoadingReport.LEFT_SIDE;

        else if (direction == Vehicle.DIRECTION_
BACKWARD)

            report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.LoadProduct(report);

            // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

            if (report.amount > 0)

                cont20WBModified = true;

        }

        //

        // Load the cont40

        //

        if (GetProcessOutput("multi_consumer_producer",
cont40ftQueue, cont40ftAsset) > 0)

        {

            spaceAvailable = cont40ftQueue.GetQueueCount();

            report = CreateLoadingReport(cont40ftQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.LoadProduct(report);

            // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

            if (report.amount > 0)

                cont40WBModified = true;

        }

        //

        // Load the gengoods

        //

        if (GetProcessOutput("multi_consumer_producer",
gengoodsQueue, gengoodsAsset) > 0)

```

```

{

    spaceAvailable = gengoodsQueue.GetQueueCount();

    report = CreateLoadingReport(gengoodsQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        goodsWBModified = true;

}

//

// Load the logs

//

    if (GetProcessOutput("multi_consumer_producer",
logsQueue, logsAsset) > 0)

    {

        spaceAvailable = logsQueue.GetQueueCount();

        report = CreateLoadingReport(logsQueue,
spaceAvailable);

        direction = vehicle.GetRelationToTrack(me, "out_
track0");

        if (direction == Vehicle.DIRECTION_FORWARD)

            report.sideFlags = LoadingReport.LEFT_SIDE;

        else if (direction == Vehicle.DIRECTION_
BACKWARD)

            report.sideFlags = LoadingReport.RIGHT_SIDE;

        vehicle.LoadProduct(report);

        // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

        if (report.amount > 0)

            logWBModified = true;

    }

```

```

//

// Load the lumber

//

if (GetProcessOutput("multi_consumer_producer",
lumberQueue, lumberAsset) > 0)

{

    spaceAvailable = lumberQueue.GetQueueCount();

    report = CreateLoadingReport(lumberQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        lumberWBModified = true;

}

//

// Load the woodchips

//

if (GetProcessOutput("multi_consumer_producer",
woodchipsQueue, woodchipsAsset) > 0)

{

    spaceAvailable = woodchipsQueue.GetQueueCount();

    report = CreateLoadingReport(woodchipsQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

```

```

// Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        woodchipWBModified = true;

}

//

// Load the water

//

if (GetProcessOutput("multi_consumer_producer",
waterQueue, waterAsset) > 0)

{

    spaceAvailable = waterQueue.GetQueueCount();

    report = CreateLoadingReport(waterQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.LoadProduct(report);

    // Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

    if (report.amount > 0)

        waterWBModified = true;

}

//

// Load the avgas

//

if (GetProcessOutput("multi_consumer_producer",
avgasQueue, avgasAsset) > 0)

{

    spaceAvailable = avgasQueue.GetQueueCount();

    report = CreateLoadingReport(avgasQueue,
spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

```

```

report.sideFlags = LoadingReport.LEFT_SIDE;

else if (direction == Vehicle.DIRECTION_
BACKWARD)

    report.sideFlags = LoadingReport.RIGHT_SIDE;

vehicle.LoadProduct(report);

// Already done something to the queue? if so, set flag so
that we don't unload it again. (just for this multi ind)

if (report.amount > 0)

    avWBModified = true;

}

}

if (itc.IsTrainCommand(vehicle.GetMyTrain(), Industry.
UNLOAD_COMMAND))

{

    // Attempt to unload everything! ;)

    //

    // Unload the crudeoil

    //

    if (GetProcessInput("multi_consumer_producer",
crudeOilQueue, crudeOilAsset) > 0)

    {

        if (!oilWBModified)

        {

            spaceAvailable = crudeOilQueue.GetQueueSpace();

            report = CreateUnloadingReport(crudeOilQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.UnloadProduct(report);

            // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

            if (oilWBRemain > 0)

```

```

oilWBRemain = oilWBRemain - report.amount;

    }

}

//

// Unload the diesel

//

if (GetProcessInput("multi_consumer_producer",
dieselQueue, dieselAsset) > 0)

{

    if (!dieselWBModified)

    {

        spaceAvailable = dieselQueue.GetQueueSpace();

        report = CreateUnloadingReport(dieselQueue,
spaceAvailable);

        direction = vehicle.GetRelationToTrack(me, "out_
track0");

        if (direction == Vehicle.DIRECTION_FORWARD)

            report.sideFlags = LoadingReport.LEFT_SIDE;

        else if (direction == Vehicle.DIRECTION_
BACKWARD)

            report.sideFlags = LoadingReport.RIGHT_SIDE;

        vehicle.UnloadProduct(report);

        // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

        if (dieselWBRemain > 0)

            dieselWBRemain = dieselWBRemain - report.amount;

    }

}

//

// Unload the petrol

//

if (GetProcessInput("multi_consumer_producer",
petrolQueue, petrolAsset) > 0)

{

    if (!petrolWBModified)

    {

        spaceAvailable = petrolQueue.GetQueueSpace();

```



```

        report = CreateUnloadingReport(petrolQueue,
spaceAvailable);

        direction = vehicle.GetRelationToTrack(me, "out_
track0");

        if (direction == Vehicle.DIRECTION_FORWARD)

            report.sideFlags = LoadingReport.LEFT_SIDE;

        else if (direction == Vehicle.DIRECTION_
BACKWARD)

            report.sideFlags = LoadingReport.RIGHT_SIDE;

        vehicle.UnloadProduct(report);

        // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

        if (petrolWBRemain > 0)

            petrolWBRemain = petrolWBRemain - report.amount;

    }

}

//

// Unload the coal

//

    if (GetProcessInput("multi_consumer_producer",
coalQueue, coalAsset) > 0)

    {

        if (!coalWBModified)

        {

            spaceAvailable = coalQueue.GetQueueSpace();

            report = CreateUnloadingReport(coalQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.UnloadProduct(report);

            // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

            if (coalWBRemain > 0)

                coalWBRemain = coalWBRemain - report.amount;

```

```

    }

}

//

// Unload the cont20

//

    if (GetProcessInput("multi_consumer_producer",
cont20ftQueue, cont20ftAsset) > 0)

    {

        if (!cont20WBModified)

        {

            spaceAvailable = cont20ftQueue.GetQueueSpace();

            report = CreateUnloadingReport(cont20ftQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.UnloadProduct(report);

            // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

            if (cont20WBRemain > 0)

                cont20WBRemain = cont20WBRemain - report.
amount;

        }

    }

//

// Unload the cont40

//

    if (GetProcessInput("multi_consumer_producer",
cont40ftQueue, cont40ftAsset) > 0)

    {

        if (!cont40WBModified)

        {

            spaceAvailable = cont40ftQueue.GetQueueSpace();

            report = CreateUnloadingReport(cont40ftQueue,

```

```

spaceAvailable);

    direction = vehicle.GetRelationToTrack(me, "out_
track0");

    if (direction == Vehicle.DIRECTION_FORWARD)

        report.sideFlags = LoadingReport.LEFT_SIDE;

    else if (direction == Vehicle.DIRECTION_
BACKWARD)

        report.sideFlags = LoadingReport.RIGHT_SIDE;

    vehicle.UnloadProduct(report);

    // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

    if (cont40WBRemain > 0)

        cont40WBRemain = cont40WBRemain - report.
amount;

    }

    }

    //

    // Unload the gengoods

    //

    if (GetProcessInput("multi_consumer_producer",
gengoodsQueue, gengoodsAsset) > 0)

    {

        if (!goodsWBModified)

        {

            spaceAvailable = gengoodsQueue.GetQueueSpace();

            report = CreateUnloadingReport(gengoodsQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.UnloadProduct(report);

            // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

            if (goodsWBRemain > 0)

```

```

        goodsWBRemain = goodsWBRemain - report.
amount;

        }

    }

    //

    // Unload the logs

    //

    if (GetProcessInput("multi_consumer_producer",
logsQueue, logsAsset) > 0)

    {

        if (!logWBModified)

        {

            spaceAvailable = logsQueue.GetQueueSpace();

            report = CreateUnloadingReport(logsQueue,
spaceAvailable);

            direction = vehicle.GetRelationToTrack(me, "out_
track0");

            if (direction == Vehicle.DIRECTION_FORWARD)

                report.sideFlags = LoadingReport.LEFT_SIDE;

            else if (direction == Vehicle.DIRECTION_
BACKWARD)

                report.sideFlags = LoadingReport.RIGHT_SIDE;

            vehicle.UnloadProduct(report);

            // Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

            if (logWBRemain > 0)

                logWBRemain = logWBRemain - report.amount;

        }

    }

    //

    // Unload the lumber

    //

    if (GetProcessInput("multi_consumer_producer",
lumberQueue, lumberAsset) > 0)

    {

        if (!lumberWBModified)

        {

            spaceAvailable = lumberQueue.GetQueueSpace();

```

```

report = CreateUnloadingReport(lumberQueue,
spaceAvailable);

direction = vehicle.GetRelationToTrack(me, "out_
track0");

if (direction == Vehicle.DIRECTION_FORWARD)

report.sideFlags = LoadingReport.LEFT_SIDE;

else if (direction == Vehicle.DIRECTION_
BACKWARD)

report.sideFlags = LoadingReport.RIGHT_SIDE;

vehicle.UnloadProduct(report);

// Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

if (lumberWBRemain > 0)

lumberWBRemain = lumberWBRemain - report.
amount;

}

}

//

// Unload the woodchips

//

if (GetProcessInput("multi_consumer_producer",
woodchipsQueue, woodchipsAsset) > 0)

{

if (!woodchipWBModified)

{

spaceAvailable = woodchipsQueue.GetQueueSpace();

report = CreateUnloadingReport(woodchipsQueue,
spaceAvailable);

direction = vehicle.GetRelationToTrack(me, "out_
track0");

if (direction == Vehicle.DIRECTION_FORWARD)

report.sideFlags = LoadingReport.LEFT_SIDE;

else if (direction == Vehicle.DIRECTION_
BACKWARD)

report.sideFlags = LoadingReport.RIGHT_SIDE;

vehicle.UnloadProduct(report);

// Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

```

```

if (woodchipWBRemain > 0)

woodchipWBRemain = woodchipWBRemain - report.
amount;

}

}

//

// Unload the water

//

if (GetProcessInput("multi_consumer_producer",
waterQueue, waterAsset) > 0)

{

if (!waterWBModified)

{

spaceAvailable = waterQueue.GetQueueSpace();

report = CreateUnloadingReport(waterQueue,
spaceAvailable);

direction = vehicle.GetRelationToTrack(me, "out_
track0");

if (direction == Vehicle.DIRECTION_FORWARD)

report.sideFlags = LoadingReport.LEFT_SIDE;

else if (direction == Vehicle.DIRECTION_
BACKWARD)

report.sideFlags = LoadingReport.RIGHT_SIDE;

vehicle.UnloadProduct(report);

// Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

if (waterWBRemain > 0)

waterWBRemain = waterWBRemain - report.amount;

}

}

//

// Unload the avgas

//

if (GetProcessInput("multi_consumer_producer",
avgasQueue, avgasAsset) > 0)

{

if (!avWBModified)

{

```

```

spaceAvailable = avgasQueue.GetQueueSpace();

report = CreateUnloadingReport(avgasQueue,
spaceAvailable);

direction = vehicle.GetRelationToTrack(me, "out_
track0");

if (direction == Vehicle.DIRECTION_FORWARD)

report.sideFlags = LoadingReport.LEFT_SIDE;

else if (direction == Vehicle.DIRECTION_
BACKWARD)

report.sideFlags = LoadingReport.RIGHT_SIDE;

vehicle.UnloadProduct(report);

// Ensure we are tracking this if we are waiting for a
way bill to complete in its entirety

if (avWBRemain > 0)

avWBRemain = avWBRemain - report.amount;

}

}

}

}

}

//

//

//

thread void MultipleMain(void)

{

Message msg;

Vehicle vehicle;

string triggerName;

wait()

{

// ?

on "Scriptlet-Enabled", "1":

{

if (!scriptletEnabled)

{

scriptletEnabled = true;

```

```

SetProcessEnabled("multi_consumer_producer", true);

}

continue;

}

// ?

on "Scriptlet-Enabled", "0":

{

if (scriptletEnabled)

{

scriptletEnabled = false;

SetProcessEnabled("multi_consumer_producer", false);

}

continue;

}

}

}

// called by the game once when a process is ready to start
(see Industry.gs)

void NotifyProcessStarted(string processName)

{

//Interface.Print("A process in the oil refinery has started.");

if (PerformProcessInput(processName))

{

// we are making the assumption that there is only one
process, "multi_consumer_producer"

PerformProcessStarted(processName);

}

else

PerformProcessCancelled(processName);

}

//

// Called by the game once when a process is ready to stop
(see Industry.gs)

//

void NotifyProcessFinished(string processName)

{

```

```

//Interface.Print("A process in the oil refinery has finished.");

processing = false;

if (!animating)
{
    PerformProcessOutput(processName);

    PerformProcessFinished(processName);

    PostMessage(me,
"GenericIndustry", "ProcessComplete", 0.0f);
}
}

//
//
//

public void Init(void)
{
    inherited();

    usePipeAnimation = false;

    useGenericViewDetails = true;

    // initialize queues

    crudeOilQueue = GetQueue("crude_oil_q");
    dieselQueue = GetQueue("diesel_q");
    petrolQueue = GetQueue("petrol_q");
    coalQueue = GetQueue("coal_q");
    cont20ftQueue = GetQueue("20ft_cont_q");
    cont40ftQueue = GetQueue("40ft_cont_q");
    gengoodsQueue = GetQueue("gen_goods_q");
    logsQueue = GetQueue("logs_q");
    lumberQueue = GetQueue("lumber_q");
    woodchipsQueue = GetQueue("woodchips_q");
    waterQueue = GetQueue("water_q");
    avgasQueue = GetQueue("av_in_q");
    crudeOilAsset = GetAsset().FindAsset("oil");
    dieselAsset = GetAsset().FindAsset("diesel");
    petrolAsset = GetAsset().FindAsset("petrol");
    coalAsset = GetAsset().FindAsset("coal");

```

```

cont20ftAsset = GetAsset().FindAsset("cont20ft");

cont40ftAsset = GetAsset().FindAsset("cont40ft");

gengoodsAsset = GetAsset().FindAsset("gengoods");

logsAsset = GetAsset().FindAsset("logs");

lumberAsset = GetAsset().FindAsset("lumber");

woodchipsAsset = GetAsset().FindAsset("woodchips");

waterAsset = GetAsset().FindAsset("water");

avgasAsset = GetAsset().FindAsset("avgas");

AddAssetToIndustryProductInfo("oil", "crude_oil_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("petrol", "petrol_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("diesel", "diesel_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("coal", "coal_q", "multi_
consumer_producer", true, false);

AddAssetToIndustryProductInfo("cont20ft", "20ft_cont_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("cont40ft", "40ft_cont_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("gengoods", "gen_goods_
q", "multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("logs", "logs_q", "multi_
consumer_producer", true, false);

AddAssetToIndustryProductInfo("lumber", "lumber_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("woodchips", "woodchips_
q", "multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("water", "water_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("avgas", "av_in_q",
"multi_consumer_producer", true, false);

AddAssetToIndustryProductInfo("oil", "crude_oil_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("petrol", "petrol_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("diesel", "diesel_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("coal", "coal_q", "multi_
consumer_producer", false);

AddAssetToIndustryProductInfo("cont20ft", "20ft_cont_q",
"multi_consumer_producer", false);

```

```

AddAssetToIndustryProductInfo("cont40ft", "40ft_cont_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("gengoods", "gen_goods_
q", "multi_consumer_producer", false);

AddAssetToIndustryProductInfo("logs", "logs_q", "multi_
consumer_producer", false);

AddAssetToIndustryProductInfo("lumber", "lumber_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("woodchips", "woodchips_
q", "multi_consumer_producer", false);

AddAssetToIndustryProductInfo("water", "water_q",
"multi_consumer_producer", false);

AddAssetToIndustryProductInfo("avgas", "av_in_q",
"multi_consumer_producer", false);

// stop animation on refinery mesh - fueling doors are shut
and not animated

StopMeshAnimation("fuelling_doors");

SetMeshAnimationFrame("fuelling_doors", 0);

MultipleMain();
}

public Requirement[] GetRequirements(void)
{
    Requirement[] ret = new Requirement[0];

    int rate = 0;

    rate = GetProcessInput("multi_consumer_producer",
GetQueue("20ft_cont_q"), GetAsset().FindAsset("cont20ft"));

    if (rate > 0)
    {
        if (cont20ftQueue.GetQueueCount() < 30 or
cont20WBRemain > 0)

        {
            ResourceRequirement req = new ResourceRequirement();

            req.resource = cont20ftQueue.GetProductFilter().
GetProducts()[0];

            // This is how many we have asked for. Wait till it is
fulfilled,

            // if we are not already waiting for a waybill to be
completed.

            req.amount = 30;

            if (cont20ftQueue.GetQueueCount() < 30 and
cont20WBRemain == 0)

```

```

cont20WBRemain = 30;

req.dst = me;

req.dstQueue = cont20ftQueue;

ret[ret.size()] = req;

}

}

rate = GetProcessInput("multi_consumer_producer",
GetQueue("40ft_cont_q"), GetAsset().FindAsset("cont40ft"));

if (rate > 0)
{
    if (cont40ftQueue.GetQueueCount() < 30 or
cont40WBRemain > 0)

    {
        ResourceRequirement req = new ResourceRequirement();

        req.resource = cont40ftQueue.GetProductFilter().
GetProducts()[0];

        // This is how many we have asked for. Wait till it is
fulfilled,

        // if we are not already waiting for a waybill to be
completed.

        req.amount = 30;

        if (cont40ftQueue.GetQueueCount() < 30 and
cont40WBRemain == 0)

        cont40WBRemain = 30;

        req.dst = me;

        req.dstQueue = cont40ftQueue;

        ret[ret.size()] = req;

    }

}

rate = GetProcessInput("multi_consumer_
producer", GetQueue("gen_goods_q"), GetAsset().
FindAsset("gengoods"));

if (rate > 0)
{
    if (gengoodsQueue.GetQueueCount() < 15 or
goodsWBRemain > 0)

    {
        ResourceRequirement req = new ResourceRequirement();

        req.resource = gengoodsQueue.GetProductFilter().

```

```

GetProducts()[0];

    // This is how many we have asked for. Wait till it is
    fulfilled,

    // if we are not already waiting for a waybill to be
    completed.

    req.amount = 40;

    if (gengoodsQueue.GetQueueCount() < 15 and
    goodsWBRemain == 0)

        goodsWBRemain = 40;

    req.dst = me;

    req.dstQueue = gengoodsQueue;

    ret[ret.size()] = req;
}
}

rate = GetProcessInput("multi_consumer_producer",
GetQueue("logs_q"), GetAsset().FindAsset("logs"));

if (rate > 0)

{

    if (logsQueue.GetQueueCount() < 25 or logWBRemain >
0)

    {

        ResourceRequirement req = new ResourceRequirement();

        req.resource = logsQueue.GetProductFilter().
GetProducts()[0];

        // This is how many we have asked for. Wait till it is
        fulfilled,

        // if we are not already waiting for a waybill to be
        completed.

        req.amount = 50;

        if (logsQueue.GetQueueCount() < 25 and logWBRemain
== 0)

            logWBRemain = 50;

        req.dst = me;

        req.dstQueue = logsQueue;

        ret[ret.size()] = req;
    }
}

rate = GetProcessInput("multi_consumer_producer",

```

```

GetQueue("lumber_q"), GetAsset().FindAsset("lumber"));

    if (rate > 0)

    {

        if (lumberQueue.GetQueueCount() < 40 or
lumberWBRemain > 0)

        {

            ResourceRequirement req = new ResourceRequirement();

            req.resource = lumberQueue.GetProductFilter().
GetProducts()[0];

            // This is how many we have asked for. Wait till it is
            fulfilled,

            // if we are not already waiting for a waybill to be
            completed.

            req.amount = 30;

            if (lumberQueue.GetQueueCount() < 40 and
lumberWBRemain == 0)

                lumberWBRemain = 30;

            req.dst = me;

            req.dstQueue = lumberQueue;

            ret[ret.size()] = req;

        }
    }

    rate = GetProcessInput("multi_consumer_producer",
GetQueue("coal_q"), GetAsset().FindAsset("coal"));

    if (rate > 0)

    {

        if (coalQueue.GetQueueCount() < 271500 or
coalWBRemain > 0) // 7 mins till empty

        {

            ResourceRequirement req = new ResourceRequirement();

            req.resource = coalQueue.GetProductFilter().
GetProducts()[0];

            // This is how many we have asked for. Wait till it is
            fulfilled,

            // if we are not already waiting for a waybill to be
            completed.

            req.amount = 814500;

            if (coalQueue.GetQueueCount() < 271500 and
coalWBRemain == 0)

```



```

        coalWBRemain = 814500;    // 15 hoppers

        req.dst = me;

        req.dstQueue = coalQueue;

        ret[ret.size()] = req;
    }
}

        rate = GetProcessInput("multi_consumer_
producer", GetQueue("woodchips_q"), GetAsset().
FindAsset("woodchips"));

        if (rate > 0)
        {
            if (woodchipsQueue.GetQueueCount() < 181500 or
woodchipWBRemain > 0) // 3 gondolas
            {
                ResourceRequirement req = new ResourceRequirement();

                req.resource = woodchipsQueue.GetProductFilter().
GetProducts()[0];

                // This is how many we have asked for. Wait till it is
fulfilled,

                // if we are not already waiting for a waybill to be
completed.

                req.amount = 544500;

                if (woodchipsQueue.GetQueueCount() < 181500 and
woodchipWBRemain == 0)

                    woodchipWBRemain = 544500;    // 9 gondolas

                req.dst = me;

                req.dstQueue = woodchipsQueue;

                ret[ret.size()] = req;
            }
        }

        rate = GetProcessInput("multi_consumer_producer",
GetQueue("crude_oil_q"), GetAsset().FindAsset("oil"));

        if (rate > 0)
        {
            if (crudeOilQueue.GetQueueCount() < 27500 or
oilWBRemain > 0) // approx 14 min to empty
            {
                ResourceRequirement req = new ResourceRequirement();

```

```

                req.resource = crudeOilQueue.GetProductFilter().
GetProducts()[0];

                // This is how many we have asked for. Wait till it is
fulfilled,

                // if we are not already waiting for a waybill to be
completed.

                req.amount = 543000;

                if (crudeOilQueue.GetQueueCount() < 27500 and
oilWBRemain == 0)

                    oilWBRemain = 543000;    // 10 tank cars

                req.dst = me;

                req.dstQueue = crudeOilQueue;

                ret[ret.size()] = req;
            }
        }

        rate = GetProcessInput("multi_consumer_producer",
GetQueue("diesel_q"), GetAsset().FindAsset("diesel"));

        if (rate > 0)
        {
            if (dieselQueue.GetQueueCount() < 232800 or
dieselWBRemain > 0) // 6 tankers
            {
                ResourceRequirement req = new ResourceRequirement();

                req.resource = dieselQueue.GetProductFilter().
GetProducts()[0];

                // This is how many we have asked for. Wait till it is
fulfilled,

                // if we are not already waiting for a waybill to be
completed.

                req.amount = 776000;

                if (dieselQueue.GetQueueCount() < 232800 and
dieselWBRemain == 0)

                    dieselWBRemain = 776000;    // 20 tankers

                req.dst = me;

                req.dstQueue = dieselQueue;

                ret[ret.size()] = req;
            }
        }

        rate = GetProcessInput("multi_consumer_producer",
GetQueue("petrol_q"), GetAsset().FindAsset("petrol"));

```

```

if (rate > 0)
{
    if (petrolQueue.GetQueueCount() < 77600 or
petrolWBRemain > 0) // 2 tankers approx 10 min
    {
        ResourceRequirement req = new ResourceRequirement();
        req.resource = petrolQueue.GetProductFilter().
GetProducts()[0];

        // This is how many we have asked for. Wait till it is
fulfilled,

        // if we are not already waiting for a waybill to be
completed.

        req.amount = 116400;

        if (petrolQueue.GetQueueCount() < 77600 and
petrolWBRemain == 0)

            petrolWBRemain = 116400;    // 3 tankers

        req.dst = me;

        req.dstQueue = petrolQueue;

        ret[ret.size()] = req;
    }
}

/* NOT BEING USED AS WE HAVE NO WAY OF
TRANSPORTING WATER YET

    rate = GetProcessInput("multi_consumer_producer",
GetQueue("water_q"), GetAsset().FindAsset("water"));

    if (rate > 0)
    {

        if (waterQueue.GetQueueCount() < 77600 or
waterWBRemain > 0) // 2 tankers approx 10 min //
VAUGHAN EDIT

        {

            ResourceRequirement req = new ResourceRequirement();

            req.resource = waterQueue.GetProductFilter().
GetProducts()[0];

            // This is how many we have asked for. Wait till it is
fulfilled,

            // if we are not already waiting for a waybill to be
completed.

            req.amount = 116400; // VAUGHAN EDIT

            if (waterQueue.GetQueueCount() < 77600 and
petrolWBRemain == 0) // VAUGHAN EDIT

```

```

        waterWBRemain = 116400;    // 3 tankers    //
VAUGHAN EDIT

        req.dst = me;

        req.dstQueue = waterQueue;

        ret[ret.size()] = req;
    }
}*/

    rate = GetProcessInput("multi_consumer_producer",
GetQueue("av_in_q"), GetAsset().FindAsset("avgas"));

    if (rate > 0)
    {

        if (avgasQueue.GetQueueCount() < 145000 or
avWBRemain > 0) // approx 25% of full

        {

            ResourceRequirement req = new ResourceRequirement();

            req.resource = avgasQueue.GetProductFilter().
GetProducts()[0];

            // This is how many we have asked for. Wait till it is
fulfilled,

            // if we are not already waiting for a waybill to be
completed.

            req.amount = 465600;    // 12 tank cars

            if (avgasQueue.GetQueueCount() < 145000 and
avWBRemain == 0)

                avWBRemain = 465600;

            req.dst = me;

            req.dstQueue = avgasQueue;

            ret[ret.size()] = req;
        }
    }

    return ret;
}

public void AppendDriverDestinations(string[] destNames,
string[] destTracks)
{

```

```
StringTable stringTable = GetAsset().GetStringTable();

    destNames[destNames.size()] = stringTable.
GetString("multi_PickupDropoff");

    destTracks[destTracks.size()] = "out_track0";
}

};
```

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Industry (Coal Mine)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

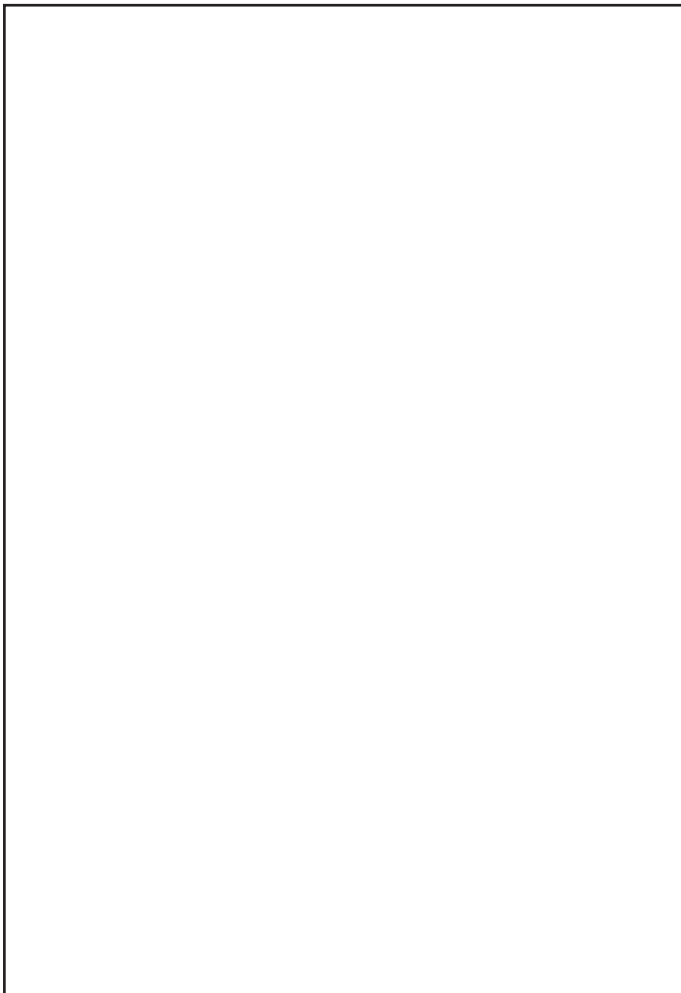
thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings



Download this asset

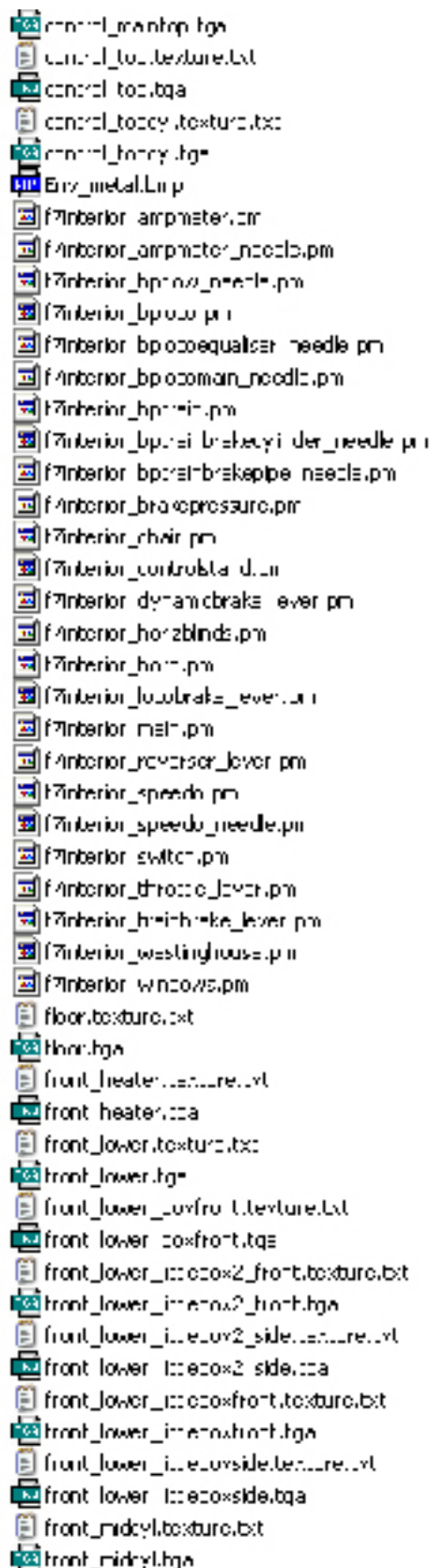
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Interior (Diesel)

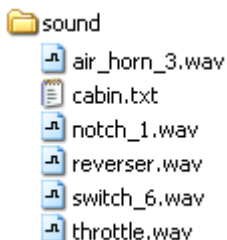
Directory Structure

A typical asset of this kind has the following File\Directory Structure:



 front_wire.png texture.txt
 front_wire.tga
 grey_metal.texture.txt
 grey_metal.tga
 horiz_line.tga
 horiz_line_alpha.bmp
 horiz_line-horiz bind_alpha texture.txt
 lh_consolefront.texture.txt
 lh_consolefront.tga
 lh_consoleside_lh.texture.txt
 lh_consoleside_lh.tga
 lh_raisededge_h.texture.txt
 lh_raisededge_h.tga
 lh_raisededge_v.texture.txt
 lh_raisededge_v.tga
 lh_raisedfloor texture.txt
 lh_raisedfloor.tga
 lh_side.texture.txt
 lh_side.tga
 lh_sidefront texture.txt
 lh_sidefront.tga
 lh_side_upper texture.txt
 lh_side_upper.tga
 needle_darkred.texture.txt
 needle_darkred.tga
 needle_red.texture.txt
 needle_red.tga
 needle_white.texture.txt
 needle_white.tga
 rear_panel.texture.txt
 rear_panel.tga
 rh_consolebevel.texture.txt
 rh_consolebevel.tga
 rh_consolefront.texture.txt
 rh_consolefront.tga
 rh_consoleside.texture.txt
 rh_consoleside.tga
 rh_raisededge_h.texture.txt
 rh_raisededge_h.tga
 rh_raisededge_v.texture.txt
 rh_raisededge_v.tga
 rh_raisedfloor.texture.txt
 rh_raisedfloor.tga
 rh_side.texture.txt
 rh_side.tga
 rh_sidefront.texture.txt
 rh_sidefront.tga
 rh_side_upper.texture.txt
 rh_side_upper.tga

 roof_rh.texture.txt
 roof_rh.tga
 rrrr.png texture.txt
 rule.tga
 speedo_face.texture.txt
 speedo_face.tga
 speedo_inside.texture.txt
 speedo_inside.tga
 speedo_rhface.texture.txt
 speedo_rhface.tga
 speedo_screw.texture.txt
 speedo_screw.tga
 speedo_top texture.txt
 speedo_top.tga
 switch_panel.png texture.txt
 switch_panel.tga
 th_rh.tga
 vert_line.tga
 vert_line-rl-hh.png
 vert_line-vert_line_alpha.texture.txt
 warninglight_off.texture.txt
 warninglight_off.tga
 westing_hose.png texture.txt
 westing_base_upper.tga
 westing_blackoxy.texture.txt
 westing_blackoxy.tga
 westing_blackoxy_top.texture.txt
 westing_blackoxy_top.tga
 westing_blacksmal.tga
 westing_blacksmal.tga
 westing_blacksurround_inside.texture.txt
 westing_blacksurround_inside.tga
 westing_blacksurround_inside.texture.txt
 westing_blacksurround_inside.tga
 westing_blacktop.texture.txt
 westing_blacktop.tga
 westing_blacktop.texture.txt
 westing_blacktop.tga
 westing_blacktop.texture.txt
 westing_blacktop.tga
 westing_blacktop.texture.txt
 westing_blacktop.tga
 westing_blacktop.texture.txt
 westing_blacktop.tga



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

ampmeter_dial_alpha.bmp, ampmeter_table_alpha.
bmp, BPgauge_backplate_alpha.bmp, BPgauge_
wires_alpha.bmp, BPloco_alpha.bmp, Env_metal.

bmp, horiz_blind_alpha.bmp, vert_blind_alpha.
bmp, ampmeter_dial.tga, ampmeter_table.tga,
black.tga, BPgauge_backplate.tga, BPgauge_dial.
tga, BPgauge_surround.tga, BPgauge_wires.tga,
BPloco_dial.tga, BPtrain_dial.tga, brass_metal.tga,
chair_back.tga, chair_base.tga, chair_lhside.tga,
chair_lhstand.tga, chair_rhside.tga, chair_rhstand.
tga, chair_seattop.tga, chair_vlowside.tga, chair_
vtopside.tga, control_maincyl.tga, control_maintop.
tga, control_top.tga, control_topcyl.tga, floor.
tga, front_heater.tga, front_lower.tga, front_lower_
boxfront.tga, front_lower_littlebox2_front.tga, front_
lower_littlebox2_side.tga, front_lower_littleboxfront.
tga, front_lower_littleboxside.tga, front_midcyl.
tga, front_wire.tga, grey_metal.tga, horiz_blind.
tga, lh_consolefront.tga, lh_consoleside_hi.tga,
lh_raisededge_h.tga, lh_raisededge_v.tga, lh_
raisedfloor.tga, lh_side.tga, lh_sidefront.tga, lh_
sideupper.tga, needle_darkred.tga, needle_red.tga,
needle_white.tga, rear_panel.tga, rh_consolebevel.
tga, rh_consolefront.tga, rh_consoleside.tga,
rh_raisededge_h.tga, rh_raisededge_v.tga,
rh_raisedfloor.tga, rh_side.tga, rh_sidefront.tga,
rh_sideupper.tga, roof_hi.tga, rope.tga, speedo_face.
tga, speedo_lhside.tga, speedo_rhside.tga, speedo_
stand.tga, speedo_top.tga, switch_panel.tga, vert_
blind.tga, warninglight_off.tga, westing_basetop.
tga, westing_blackcyl.tga, westing_blackcyltop.tga,
westing_blacksmallcyl.tga, westing_blacksurround_
lhside.tga, westing_blacksurround_rhside.tga,
westing_blacktop.tga, westing_bolt.tga, westing_
extrusion.tga, westing_frontwrap.tga, westing_
greencyl.tga, westing_greenmidtop.tga, westing_
greentop.tga, westing_maintop.tga, westing_midcyl_
hi.tga, westing_midtop2.tga, westing_midtop.tga,
westing_midtoptop.tga, westing_tallgreencyl.tga,
westing_topcyl_hi.tga, westing_topsmallcyl.tga,
wheelslip.tga, windows_lhfront.tga, windows_lhside.
tga, windows_lhsidedoor.tga, windows_rhfront.
tga, windows_rhside.tga, windows_rhsidedoor.
tga, windscreen_darktile.tga, windscreen_mid.tga,
windscreen_outeredge.tga, windscreen_surroundtile.
tga, windscreen_tile.tga, ampmeter_dial-ampmeter_
dial_alpha.texture.txt, ampmeter_table-ampmeter_
table_alpha.texture.txt, black.texture.txt, BPgauge_
backplate-BPgauge_backplate_alpha.texture.
txt, BPgauge_dial.texture.txt, BPgauge_surround.
texture.txt, BPgauge_wires-BPgauge_wires_alpha.
texture.txt, Ploco_dial-BPloco_alpha.texture.txt,
BPtrain_dial-BPloco_alpha.texture.txt, brass_metal.
texture.txt, chair_back.texture.txt, chair_base.texture.
txt, chair_lhside.texture.txt, chair_lhstand.texture.
txt, chair_rhside.texture.txt, chair_rhstand.texture.
txt, chair_seattop.texture.txt, chair_vlowside.texture.
txt, chair_vtopside.texture.txt, control_maincyl.
texture.txt, control_maintop.texture.txt, control_top.
texture.txt, control_topcyl.texture.txt, floor.texture.txt,
front_heater.texture.txt, front_lower.texture.txt, front_
lower_boxfront.texture.txt, front_lower_littlebox2_
front.texture.txt, front_lower_littlebox2_side.texture.
txt, front_lower_littleboxfront.texture.txt, front_lower_
littleboxside.texture.txt, front_midcyl.texture.txt,
front_wire.texture.txt, grey_metal.texture.txt, horiz

kind	"needle"
mesh needle.pm"	"f7interior_bplocomain_
att	"a.bplocomain_needle"
limits	0,1000
att-parent	"default"
bptrainbrakecylinder_needle	
kind	"needle"
mesh bptrainbrakecylinder_needle.pm"	"f7interior_
att needle"	"a.bptrainbrakecylinder_
limits	0,1000
att-parent	"default"
bptrainbrakepipe_needle	
kind	"needle"
mesh bptrainbrakepipe_needle.pm"	"f7interior_
att needle"	"a.bptrainbrakepipe_
limits	0,1000
att-parent	"default"
speedo_needle	
kind	"needle"
mesh needle.pm"	"f7interior_speedo_
att	"a.speedo_needle"
limits	0,58
att-parent	"default"
dynamicbrake_lever	
kind	"lever"
mesh dynamicbrake_lever.pm"	"f7interior_
att	"a.dynamicbrake_lever"
limits	0,2
angles	0,0.94
notches	0,0.5,1
notchheight	1,1,1

att-parent	"default"
independantbrake_lever	
kind	"lever"
mesh lever.pm"	"f7interior_locobrake_
att	"a.locobrake_lever"
limits	0,32
angles	0.94,0
notches	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1
notchheight	1,2,2,2,2,2,2,2,1
mousespeed	-1
att-parent	"default"
reverser_lever	
kind	"lever"
mesh lever.pm"	"f7interior_reverser_
att	"a.reverser_lever"
limits	0,2
angles	0,-0.471239
notches	0,0.5,1
notchheight	1,1,1
att-parent	"default"
throttle_lever	
kind	"lever"
mesh pm"	"f7interior_throttle_lever.
att	"a.throttle_lever"
limits	0,8
angles	-0.471,0
notches	0,0.125,0.25,0.375,0.5,0.625,0.75,0.875,1
notchheight	1,1,1,1,1,1,1,1,1
mousespeed	-1
att-parent	"default"
trainbrakelap_lever	
kind	"lever"

mesh	"f7interior_trainbrake_lever.pm"
att	"a.trainbrake_lever"
limits	0,4
angles	0.94,0
notches	0,0.25,0.5,0.75,1
notchheight	1,1,1,1,1
mousespeed	-1
att-parent	"default"
horn	
kind	"pullrope"
mesh	"f7interior_horn.pm"
att	"a.horn"
limits	0,1
angles	0.1,0
notches	0,1
notchheight	0,0
mousespeed	-1
att-parent	"default"
1	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch0"
limits	0,1
angles	0,2
mousespeed	-1
radius	0.2
att-parent	"default"
2	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch1"
limits	0,1
angles	0,2
mousespeed	-1

radius	0.2
att-parent	"default"
3	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch2"
limits	0,1
angles	0,2
mousespeed	-1
radius	0.2
att-parent	"default"
4	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch3"
limits	0,1
angles	0,2
mousespeed	-1
radius	0.2
att-parent	"default"
5	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch4"
limits	0,1
angles	0,2
mousespeed	-1
radius	0.2
att-parent	"default"
6	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch5"
limits	0,1
angles	0,2

mousespeed	-1
radius	0.2
att-parent	"default"
light_switch	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch6"
limits	0,1
angles	0,2
notches	0,1
notchheight	0,0
mousespeed	-1
radius	0.2
att-parent	"default"
8	
kind	"lever"
mesh	"f7interior_switch.pm"
att	"a.switch7"
limits	0,1
angles	0,2
mousespeed	-1
radius	0.2
att-parent	"default"
ampmetermesh pm"	"f7interior_ampmeter. pm"
bplocomesh	"f7interior_bploco.pm"
bptrain	
mesh	"f7interior_bptrain.pm"
brakepressure	
mesh brakepressure.pm"	"f7interior_ brakepressure.pm"
chair	
mesh	"f7interior_chair.pm"
controlstand	
mesh pm"	"f7interior_controlstand. pm"

horizblinds	
mesh pm"	"f7interior_horizblinds. pm"
interior_main	
mesh	"f7interior_main.pm"
speedo	
mesh	"f7interior_speedo.pm"
westinghouse	
mesh westinghouse.pm"	"f7interior_ westinghouse.pm"
windows	
mesh	"f7interior_windows.pm"
opacity	0
wheelslip_light	
kind	"light"
mesh	"wheelslip.pm"
att	"none"
att-parent	"default"
default	
mesh	"f7interior_main.pm"
auto-create	1
cameralist	
camera0	0.737, 0.079,0.547,1.397,-0.321
camera1	1.014, 0.146,0.527,-0.173,0.032
camera2	-0.797, 0.476,0.547,0.057,-0.085
camera3	-0.977, 0.907,0.72,0.191,-0.452
camera4	-0.737, 0.079,0.547,-1.397,-0.321
camera5	-0.797, 0.476,0.547,0.057,-0.085
thumbnails	
0	
image	"thumb.jpg"
width	240

height	180

Download this asset

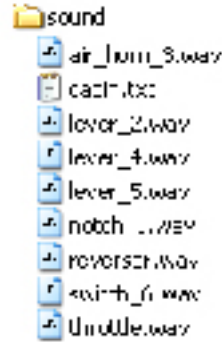
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Interior (Electric)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:





Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

cabin.txt - A text file mapping sounds to their respective interior elements.

air_horn_3.wav, lever_2.wav, lever_4.wav, lever_5.wav, notch_1.wav, reverser.wav, switch_6.wav, throttle.wav - The sound files associated with the interior elements.

exting_handle_op.bmp, phonecord_op.bmp, wheelslip_op.bmp, ammeter0_1.tga, ammeter-2_3.tga, ball.tga, black.tga, bonnet.tga, chair_base.tga, dash fr left.tga, dash fr mid.tga, dash fr right.tga, dash fr side.tga, dash front.tga, dash right edge.tga, dash sideface2.tga, dash sideface.tga, dashtop_main.tga, dial_green_large.tga, dial_green_small.tga, dial_ring.tga, dial_ring_small.tga, dial_white.tga, exting_body.tga, exting_handle.tga, front switches.tga, large_pressure.tga, lever base2.tga, lever base.tga, lever handle1.tga, leverstaff.tga, meters.tga, needle_red.tga, needle_white.tga, peddle.tga, peddle_base front.tga, peddle_base sides.tga, pent_base.tga, pent_lever side.tga, pent_lever top.tga, phone base.tga, phone.tga, phonecord.tga, red_ring.tga, ring.tga, seat.tga, silver_metal.tga, small_pressure1.tga, small_pressure2.tga, speakerpanel.tga, speedo1.tga, speedo2.tga, switch red.tga, switch white.tga, switchpanel_right.tga, tgv_int1.tga, tgv_int2.tga, tgv_int3.tga, tract front.tga, tract left.tga, tract right.tga, tract_base.tga, tract_top.tga, voltmeter0_15.tga, voltmeter0_30.tga, wheelslip.tga, window.tga, yellow.tga, ammeter0_1.texture.txt, ammeter-2_3.texture.txt, ball.texture.txt, black.texture.txt, chair_base.texture.txt, dash fr left.texture.txt, dash fr mid.texture.txt, dash fr right.texture.txt, dash fr side.texture.txt, dash front.texture.txt, dash right edge.texture.txt, dash sideface2.texture.txt, dash sideface.texture.txt, dashtop_main.texture.txt, dial_green_large.texture.txt, dial_green_small.texture.txt, dial_ring.texture.txt, dial_ring_small.texture.txt, dial_white.texture.txt, exting_body.texture.txt, exting_handle_exting_handle_op.texture.txt, front switches.texture.txt, large_pressure.texture.txt, lever base2.texture.txt, lever base.texture.txt, lever handle1.texture.txt, leverstaff.texture.txt, meters.texture.txt, needle_red.

texture.txt, needle_white.texture.txt, peddle.texture.txt, peddle_base front.texture.txt, peddle_base sides.texture.txt, pent_base.texture.txt, pent_lever side.texture.txt, pent_lever top.texture.txt, phone base.texture.txt, phone.texture.txt, phonecord-phonecord_op.texture.txt, red_ring.texture.txt, ring.texture.txt, seat.texture.txt, silver_metal.texture.txt, small_pressure1.texture.txt, small_pressure2.texture.txt, speakerpanel.texture.txt, speedo1.texture.txt, speedo2.texture.txt, switch red.texture.txt, switch white.texture.txt, switchpanel_right.texture.txt, tgv_int1.texture.txt, tgv_int2.texture.txt, tgv_int3.texture.txt, tract front.texture.txt, tract left.texture.txt, tract right.texture.txt, tract_base.texture.txt, tract_top.texture.txt, voltmeter0_15.texture.txt, voltmeter0_30.texture.txt, wheelslip-wheelslip_op.texture.txt, window.texture.txt, yellow.texture.txt - The texture files used by the progressive meshes of this interior.

bar_meter_needle.pm, brake.pm, gen_tgv_cab.pm, lever_handle.pm, light_switch.pm, pantograph_lever.pm, pressure_needle_lge.pm, pressure_needle_lge_red.pm, pressure_needle_sml.pm, reverser.pm, speedo_needle1.pm, speedo_needle2.pm, switch.pm, traction.pm, wheelslip.pm, windows.pm - The progressive mesh components used to create the interior asset. More information on modelling interior assets can be found in [Chapter 8](#) of this document.

File Listings

config.txt	
kuid	<kuid:56113:1016>
trainz-build	2.5
category-class	"ZI"
category-region	"00"
category-era	"1990s;2000s;2010s"
username	"testElectricInterior"
kind	"interior"
cameradefault	2
description	"Test electric interior. Based on the TGV interior."
cameralist	
camera0	0.773,0.671,0.2,1.566,-0.096
camera1	0.583,0.35,0.247,0,-0.352
camera2	0.479,0,0.148,0,0
camera3	-0.69,-0.017,0.17,0.242,-0.185
camera4	-0.773,0.671,0.2,-

1.566,-0.096	
camera5	0.6,0.35,0.17,0,0
mesh-table	
pantograph_lever	
kind	"lever"
mesh	"pantograph_lever.pm"
att	"a.pantograph_lever"
limits	0,1
angles	0,1
notches	0,1
notchheight	3,3
radius	0.16
att-parent	"default"
horn	
kind	"lever"
mesh	"lever_handle.pm"
att	"a.horn"
limits	0,1
angles	0,-0.45
notches	0,1
notchheight	3,3
radius	0.16
mousespeed	-1
att-parent	"default"
independantbrake_lever	
kind	"lever"
mesh	"lever_handle.pm"
att	"a.ind_brake_lever"
limits	0,32
angles	0,-0.45
notches	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1
notchheight	1,2,2,2,2,2,2,2,2,1
radius	0.15
mousespeed	-1

config.txt						limits	0,1
kuid	<kuid:56113:1016>					angles	0,-0.45
trainz-build	2.5					notches	0,1
category-class	"ZI"					notchheight	3,3
category-region	"00"					radius	0.16
category-era	"1990s;2000s;2010s"					mousespeed	-1
username	"testElectricInterior"					att-parent	"default"
kind	"interior"					independantbrake_lever	
cameradefault	2					kind	"lever"
description	"Test electric interior. Based on the TGV interior."					mesh	"lever_handle.pm"
						att	"a.ind_brake_lever"
cameralist						limits	0,32
camera0		0.773,0.671,0.2,1.566,-0.096				angles	0,-0.45
camera1		0.583,0.35,0.247,0,-0.352				notches	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1
camera2		0.479,0,0.148,0,0				notchheight	1,2,2,2,2,2,2,2,1
camera3		-0.69,-0.017,0.17,0.242,-0.185				radius	0.15
camera4		-0.773,0.671,0.2,-1.566,-0.096				mousespeed	-1
camera5		0.6,0.35,0.17,0,0				att-parent	"default"
mesh-table						reverser_lever	
pantograph_lever						kind	"lever"
kind	"lever"					mesh	"reverser.pm"
mesh	"pantograph_lever.pm"					att	"a.reverser"
att	"a.pantograph_lever"					limits	0,2
limits	0,1					angles	0,-0.471239
angles	0,1					notches	0,0.5,1
notches	0,1					notchheight	1,1,1
notchheight	3,3					att-parent	"default"
radius	0.16					throttle_lever	
att-parent	"default"					kind	"lever"
horn						mesh	"traction.pm"
kind	"lever"					att	"a.traction"
mesh	"lever_handle.pm"					limits	0,32
att	"a.horn"					angles	-0.75,0.75
						notches	0,0.0303,0.0606,0.0909,0.1212,0.1515,0.1818,0.2121,0.2424,0.2727,0.3

03,0.3333,0.3636,0.3939,0.4242,0.4545,0.4848,0.5151,0.5454,0.5757,0.606,0.6363,0.6666,0.6969,0.7272,0.7575,0.7878,0.8181,0.8484,0.8787,0.909,0.9393,1	
notchheight	1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,1
radius	0.35
att-parent	"default"
voltmeter_line	
kind	"needle"
mesh	"bar_meter_needle.pm"
att	"a.voltmeter_line"
limits	0,125
value	75
angles	0,0.6
att-parent	"default"
ampmeter_needle	
kind	"needle"
mesh	"bar_meter_needle.pm"
att	"a.ampmeter_motor1"
limits	0,1500
angles	0,0.6
att-parent	"default"
ampmeter2_needle	
kind	"needle"
mesh	"bar_meter_needle.pm"
att	"a.ampmeter_motor2"
limits	0,1500
angles	0,0.6
att-parent	"default"
ampmeter_brake	
kind	"needle"
mesh	"bar_meter_needle.pm"
att	"a.ampmeter_brake"
limits	0,1000
value	200
angles	0,0.6

att-parent	"default"
voltmeter_battery	
kind	"needle"
mesh	"bar_meter_needle.pm"
att	"a.voltmeter_battery"
limits	0,125
value	72
angles	0,0.6
att-parent	"default"
trainbrake_lever	
kind	"lever"
mesh	"brake.pm"
att	"a.train_brake_lever"
limits	0,4
angles	-0.75,0.35
notches	0,0.25,0.27,0.29,0.31,0.33,0.35,0.37,0.39,0.41,0.43,0.45,0.47,0.49,0.5,0.75,1
notchheight	1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,1,1,1
radius	0.15
mousespeed	-1
att-parent	"default"
bplocomain_needle	
kind	"needle"
mesh	"pressure_needle_lge.pm"
att	"a.res_pressure_needle"
limits	0,1000
att-parent	"default"
bploco_equaliser	
kind	"needle"
mesh	"pressure_needle_lge_red.pm"
att	"a.res_pressure_needle"
limits	0,1000
att-parent	"default"
bptrainbrakepipe_needle	
kind	"needle"

mesh	"pressure_needle_sml.pm"
att	"a.brake_cyl_pressure"
limits	0,1000
att-parent	"default"
no3pipe_needle	
kind	"needle"
mesh	"pressure_needle_sml.pm"
att	"a.ind_brake_pressure"
limits	0,1000
att-parent	"default"
speedo_needle2	
kind	"needle"
mesh	"speedo_needle1.pm"
att	"a.speedo_needle1"
limits	0,50
att-parent	"default"
speedo_needle	
kind	"needle"
mesh	"speedo_needle2.pm"
att	"a.speedo_needle2"
limits	0,186
att-parent	"default"
wheelslip_light	
kind	"light"
mesh	"wheelslip.pm"
att	"none"
att-parent	"default"
switch0	
kind	"lever"
mesh	"switch.pm"
att	"a.switch0"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0

att-parent	"default"
switch1	
kind	"lever"
mesh	"switch.pm"
att	"a.switch1"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch2	
kind	"lever"
mesh	"switch.pm"
att	"a.switch2"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch3	
kind	"lever"
mesh	"switch.pm"
att	"a.switch3"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch4	
kind	"lever"
mesh	"switch.pm"
att	"a.switch4"
limits	0,1
angles	0,1
notches	0,1

notchheight	0,0
att-parent	"default"
switch5	
kind	"lever"
mesh	"switch.pm"
att	"a.switch5"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch6	
kind	"lever"
mesh	"switch.pm"
att	"a.switch6"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch7	
kind	"lever"
mesh	"switch.pm"
att	"a.switch7"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch8	
kind	"lever"
mesh	"switch.pm"
att	"a.switch8"
limits	0,1
angles	0,1

notches	0,1
notchheight	0,0
att-parent	"default"
switch9	
kind	"lever"
mesh	"switch.pm"
att	"a.switch9"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch10	
kind	"lever"
mesh	"switch.pm"
att	"a.switch10"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch11	
kind	"lever"
mesh	"switch.pm"
att	"a.switch11"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch12	
kind	"lever"
mesh	"switch.pm"
att	"a.switch12"
limits	0,1

angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch13	
kind	"lever"
mesh	"switch.pm"
att	"a.switch13"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch14	
kind	"lever"
mesh	"switch.pm"
att	"a.switch14"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch15	
kind	"lever"
mesh	"switch.pm"
att	"a.switch15"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch16	
kind	"lever"
mesh	"switch.pm"
att	"a.switch16"

limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch17	
kind	"lever"
mesh	"switch.pm"
att	"a.switch17"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch18	
kind	"lever"
mesh	"switch.pm"
att	"a.switch18"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch19	
kind	"lever"
mesh	"switch.pm"
att	"a.switch19"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
switch20	
kind	"lever"
mesh	"switch.pm"

att	"a.switch20"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
light_switch	
kind	"lever"
att	"a.switch21"
mesh	"light_switch.pm"
limits	0,1
angles	0,1
notches	0,1
notchheight	0,0
att-parent	"default"
windows	
mesh	"windows.pm"
opacity	0
default	
mesh	"gen_tgv_cab.pm"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

cabin.txt

```
* switch switch_6.wav
dynamicbrake_lever change-notch notch_1.wav
independantbrake_lever change-notch notch_1.wav
reverser_lever change-notch lever_2.wav
throttle_lever change-notch lever_5.wav
trainbrakelap_lever change-notch notch_1.wav
pantograph_lever change-notch lever_4.wav
trainbrake_lever change-notch notch_1.wav
```

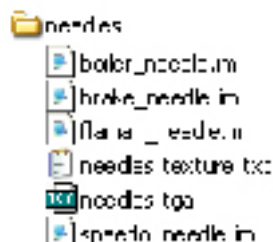
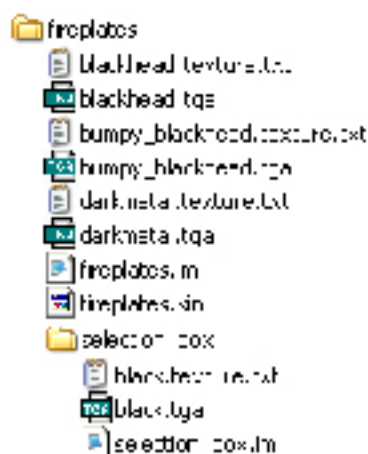
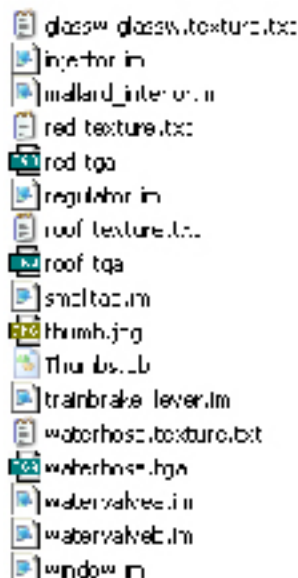
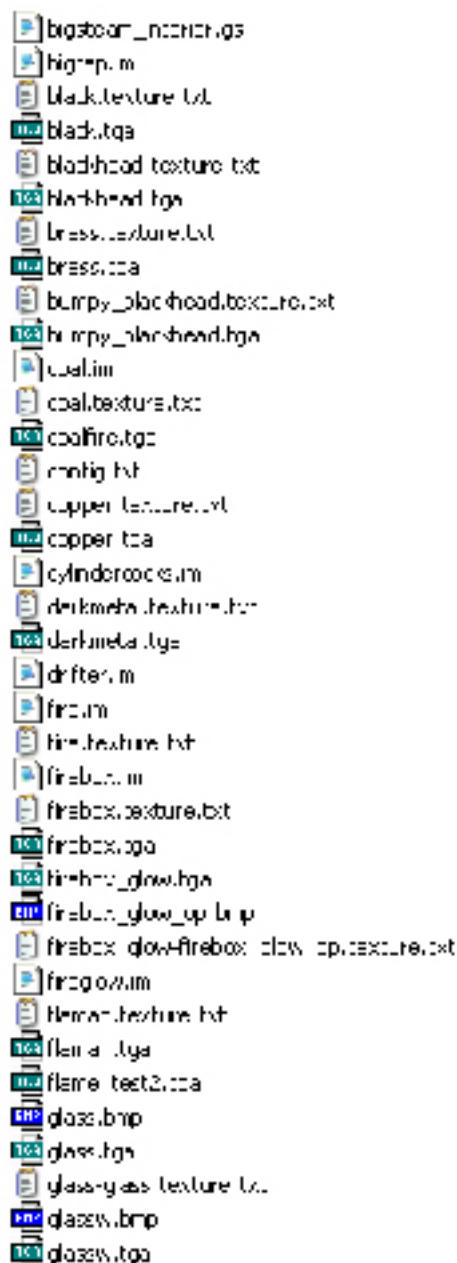
Download this asset

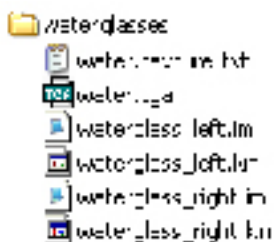
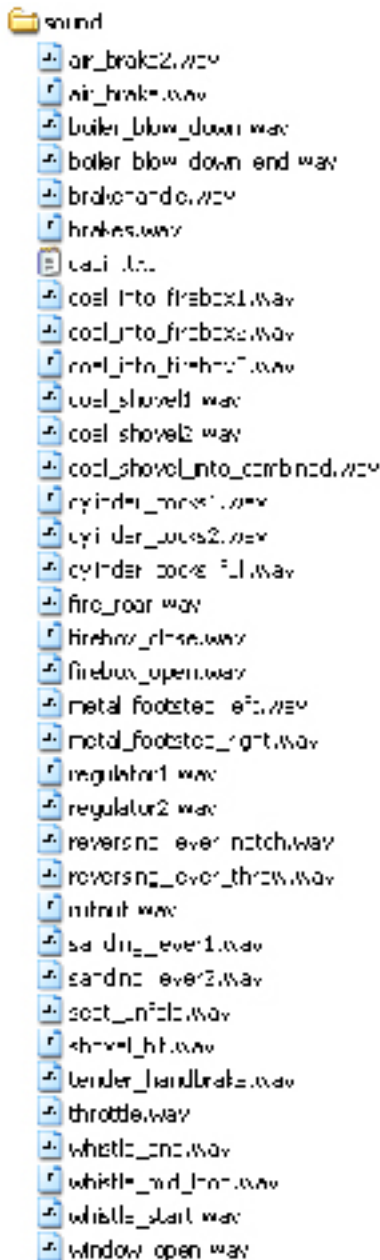
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Interior (Steam)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:





txt, copper.texture.txt, darkmetal.texture.txt, fire.texture.txt, firebox.texture.txt, firebox_glow-firebox_glow_op.texture.txt, flaman.texture.txt, glass-glass.texture.txt, glassw-glassw.texture.txt, red.texture.txt, roof.texture.txt, waterhose.texture.txt, blackhead.texture.txt, blackhead.tga, bumpy_blackhead.texture.txt, bumpy_blackhead.tga, darkmetal.texture.txt, darkmetal.tga, water.texture.txt, water.tga, needles.tga, needles.texture.txt - The texture files used by the progressive meshes of this interior.

fireplates.kin, waterglass_left.kin, waterglass_right.kin - The animation files used to operate certain elements of the interior, ie the fireplate doors opening and closing.

bigsteam_interior.gs - Script file detailing some functionality of the interior asset, in this case that no animated fireman is present in the interior.

bigtap.im, coal.im, cylindercocks.im, drifter.im, fire.im, firebox.im, fireglow.im, injector.im, mallard_interior.im, regulator.im, smalltap.im, trainbrake_lever.im, watervalvea.im, watervalveb.im, window.im, selection_box.im, waterglass_left.im, waterglass_right.im, boiler_needle.im, brake_needle.im, flaman_needle.im, speedo_needle.im, fireplates.im - The indexed mesh components used to create the interior asset. More information on modelling interior assets can be found in [Chapter 8](#) of this document.

Cabin.txt - A text file mapping sounds to their respective interior elements.

air_brake2.wav, air_brake.wav, boiler_blow_down.wav, boiler_blow_down_end.wav, brakehandle.wav, brakes.wav, coal_into_firebox1.wav, coal_into_firebox2.wav, coal_into_firebox3.wav, coal_shovel1.wav, coal_shovel2.wav, coal_shovel_into_combined.wav, cylinder_cocks1.wav, cylinder_cocks2.wav, cylinder_cocks_full.wav, fire_roar.wav, firebox_close.wav, firebox_open.wav, metal_footstep_left.wav, metal_footstep_right.wav, regulator1.wav, regulator2.wav, reversing_lever_notch.wav, reversing_lever_throw.wav, rutnut.wav, sanding_lever1.wav, sanding_lever2.wav, seat_unfold.wav, shovel_hit.wav, tender_handbrake.wav, throttle.wav, whistle_end.wav, whistle_mid_loop.wav, whistle_start.wav, window_open.wav - The sound files associated with the interior elements.

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

firebox_glow_op.bmp, glass.bmp, glassw.bmp, black.tga, blackhead.tga, brass.tga, bumpy_blackhead.tga, coalfire.tga, copper.tga, darkmetal.tga, firebox.tga, firebox_glow.tga, flaman.tga, flame_test2.tga, glass.tga, glassw.tga, red.tga, roof.tga, waterhose.tga, black.texture.txt, blackhead.texture.txt, brass.texture.txt, bumpy_blackhead.texture.txt, coal.texture.

File Listings

config.txt			
kuid	<kuid:56113:1015>	kind	"lever"
kind	"interior"	mesh	"trainbrake_lever.im"
username	"testSteamInterior"	att	"a.trainbrake_lever"
trainz-build	2.5	att-parent	"default"
cameradefault	0	auto-create	1
category-class	"ZI"	limits	0,4
category-region	"00"	angles	0,0.94
category-era	"1940s;1950s;1960s;1970s;1980s"	notches	0,0.25,0.5,0.75,1
script	"bigsteam_interior.gs"	notchheight	1,1,1,1,1
class	"Bigsteam_Interior"	mousespeed	1
cameralist		firebox	
camera0	1,0.9,0.55,0,0	kind	"firebox"
camera1	0.5,0.9,0.55,0,0	mesh	"firebox.im"
camera2	0,0.9,0.55,0,0	auto-create	1
camera3	-0.5,0.9,0.55,0,0	att	"a.origin"
camera4	-1,0.9,0.55,0,0	att-parent	"default"
camera5	1.1,0.8,0,0,0	test-collisions	0
mesh-table		fire	
default		mesh	"fire.im"
mesh	"mallard_interior.im"	auto-create	1
auto-create	1	light	0
regulator		test-collisions	0
kind	"lever"	coal	
mesh	"regulator.im"	mesh	"coal.im"
att	"a.regulator"	auto-create	1
att-parent	"default"	light	0
auto-create	1	test-collisions	0
limits	0,1	fireglow	
angles	0.24,-0.24	mesh	"fireglow.im"
radius	0.35	auto-create	1
mousespeed	-1	light	0
mousemode	"exact"	test-collisions	0
trainbrakelap_lever		water_injector_0	
		mesh	"injector.im"
		auto-create	1

att	"a.injector0"
att-parent	"default"
kind	"lever"
mousespeed	1
water_injector_1	
mesh	"injector.im"
auto-create	1
att	"a.injector1"
att-parent	"default"
kind	"lever"
mousespeed	1
fire_plates	
mesh	"fireplates/fireplates.im"
anim	"fireplates/fireplates.kin"
auto-create	1
kind	"animated-lever"
test-collisions	0
notches	0,1
notchheight	1,1
limits	0,1
fire_plates-collision-box	
mesh selection_box.im"	"fireplates/selection_box/"
att-parent	"fire_plates"
att	"a.selection_box"
auto-create	1
kind	"collision-proxy"
opacity	0
collision-parent	"fire_plates"
waterglass_right	
mesh im"	"waterglasses/waterglass_right."
anim kin"	"waterglasses/waterglass_right."
auto-create	1
limits	0,100

kind	"animated-dial"
waterglass_left	
mesh im"	"waterglasses/waterglass_left."
anim kin"	"waterglasses/waterglass_left."
auto-create	1
limits	0,100
kind	"animated-dial"
reverser	
mesh	"reverser/reverser.im"
anim	"reverser/reverser.kin"
auto-create	1
kind	"animated-lever"
test-collisions	0
limits	-1,1
mousespeed	-2
reverser-collision-box	
mesh selection_box.im"	"reverser/selection_box/"
att-parent	"reverser"
att	"a.selection_box"
auto-create	1
kind	"collision-proxy"
opacity	0
collision-parent	"reverser"
boiler_needle	
kind	"needle"
mesh	"needles/boiler_needle.im"
att	"a.boiler_pressure1"
limits	0,1902
att-parent	"default"
auto-create	1
boiler_needle1	
kind	"needle"
mesh	"needles/boiler_needle.im"

att	"a.boiler_pressure2"	water_valve__	
limits	0,1902	mesh	"watervalueA.im"
att-parent	"default"	auto-create	1
auto-create	1	att	"a.watervalue_1"
speedo_needle		att-parent	"default"
kind	"needle"	kind	"lever"
mesh	"needles/speedo_needle.im"	mousemode	"exact"
att	"a.speedo"	limits	0,1
att-parent	"default"	angles	0,-0.5
auto-create	1	water_valve__	
limits	0,49	mesh	"watervalueB.im"
bptrainbrakepipe_needle		auto-create	1
kind	"needle"	att	"a.watervalue_2"
mesh	"needles/brake_needle.im"	att-parent	"default"
att	"a.trainbrake_needle"	kind	"lever"
att-parent	"default"	mousemode	"exact"
auto-create	1	limits	0,1
limits	0,1970	angles	0,0.5
bplocomain_needle		mousespeed	-1
kind	"needle"	water_valve__	
mesh	"needles/brake_needle.im"	mesh	"watervalueB.im"
att	"a.mainres_needle"	auto-create	1
att-parent	"default"	att	"a.watervalue_3"
auto-create	1	att-parent	"default"
angles	0,-3.14	kind	"lever"
limits	0,890	mousemode	"exact"
water_valve_		limits	0,1
mesh	"watervalueA.im"	angles	0,0.5
auto-create	1	mousespeed	-1
att	"a.watervalue_0"	water_valve__	
att-parent	"default"	mesh	"watervalueB.im"
kind	"lever"	auto-create	1
mousemode	"exact"	att	"a.watervalue_4"
limits	0,1	att-parent	"default"
angles	0,-0.5	kind	"lever"

mousemode	"exact"
limits	0,1
angles	0,-0.5
mousespeed	-1
water_valve_____	
mesh	"watervalueB.im"
auto-create	1
att	"a.watervalue_5"
att-parent	"default"
kind	"lever"
mousemode	"exact"
limits	0,1
angles	0,-0.5
mousespeed	-1

mesh	"drifter.im"
auto-create	1
att	"a.drifter"
att-parent	"default"
kind	"lever"
mousemode	"exact"
limits	0,1
angles	0,-0.5
mousespeed	-1
driver_window	
mesh	"window.im"
auto-create	1
att	"a.windowA"
att-parent	"default"
limits	0,1
angles	0,-0.012
notches	0,1
notchheight	1,1
kind	"lever"
fireman_window	

mesh	"window.im"
auto-create	1
att	"a.windowB"
att-parent	"default"
limits	0,1
angles	0,0.012
notches	0,1
notchheight	1,1
kind	"lever"
cylinder_cocks	
mesh	"cylindercocks.im"
auto-create	1
att	"a.cylindercocks"
att-parent	"default"
kind	"lever"
mousemode	"exact"
limits	0,1
angles	0,-0.5

mesh	"watervalueA.im"
auto-create	1
att	"a.steamvalve"
att-parent	"default"
kind	"lever"
mousemode	"exact"
limits	0,1
angles	0,-0.5
speedo_needle1	
kind	"needle"
mesh	"needles/flaman_needle.im"
att	"a.flaman"
att-parent	"default"
auto-create	1
limits	0,85

mesh	"smalltap.im"
auto-create	1
att	"a.steamvalve0"
att-parent	"default"
kind	"lever"
mousespeed	1
mesh	"bigtap.im"
auto-create	1
att	"a.steamvalve1"
att-parent	"default"
kind	"lever"
mousespeed	1
blower	
mesh	"bigtap.im"
auto-create	1
att	"a.steamvalve2"
att-parent	"default"
kind	"lever"
mousespeed	1
mesh	"smalltap.im"
auto-create	1
att	"a.steamvalve3"
att-parent	"default"
kind	"lever"
mousespeed	1
mesh	"bigtap.im"
auto-create	1
att	"a.steamvalve4"
att-parent	"default"
kind	"lever"
mousespeed	1
thumbnails	

0	
image	"thumb.jpg"
width	240
height	180

```

bigsteam_interior.gs
include "DefaultSteamCabin.gs"
class Bigsteam_Interior isclass DefaultSteamCabin
{
    public void Init(void)
    {
        inherited();
        hasAnimatedFireman = false;
    }
};

```

```

cabin.txt
* switch switch_6.wav
dynamicbrake_lever change-notch notch_1.wav
independantbrake_lever change-notch airbrake2.wav
reverser lever-low reversing_lever_notch.wav
reverser lever-high reversing_lever_throw.wav
fire_plates lever-low firebox_open.wav
fire_plates lever-high firebox_close.wav
fire_plates looping fire_roar.wav
regulator lever-high regulator2.wav
regulator lever-low THROTTLE.wav
sanding_lever lever-low sanding_lever1.wav
sanding_lever lever-high sanding_lever2.wav
blowdown looping boiler_blow_down.wav
cylinder_cocks change-notch cylinder_cocks_full.wav
seat0 change-notch seat_unfold.wav
seat1 change-notch seat_unfold.wav
whistle_lever looping whistle_mid_loop.wav
trainbrakelap_lever change-notch air_brake2.wav

```


pantograph_lever change-notch lever_4.wav
trainbrake_lever change-notch air_brake.wav
driver_window change-notch seat_unfold.wav
fireman_window change-notch seat_unfold.wav

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Library

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

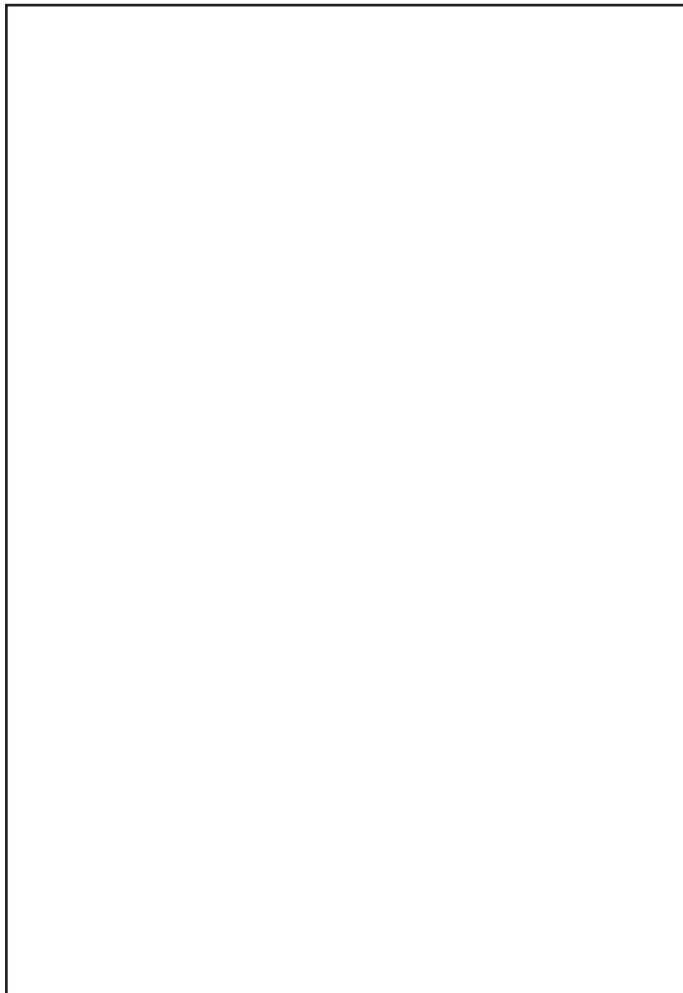
thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings



Download this asset

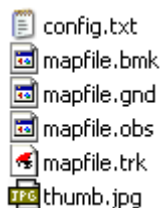
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Map

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

mapfile.bmk - The bookmark file, storing all the bookmarks you've saved in surveyor.

mapfile.gnd - The "ground file" storing information about the topology of your map.

mapfile.obs - The "objects" file storing information about the objects placed in your map.

mapfile.trk - The track layout file. This file can be opened in a 3rd party viewer such as Trainzmap.

File Listings

config.txt

region	<kuid:-1:7801>
kuid	<kuid:56113:1244>
kind	"map"
username	"testMap"
workingscale	0
workingunits	0
water	<kuid:-1:6342>
trainz-build	2.5
category-class	"YM"
carrate	0
category-region	"00"
category-era	"2000s"
description	"A test map. Generated in Trainz and edited in CCP."
world-origin	

latitude	27,28,-1
longitude	153,2,1
altitude	0
string-table	
atsf_f7a_1	"ATSF F7A 1"
kuid-table	
0	<kuid:-1:6270>
1	<kuid:-1:101452>
2	<kuid:-1:110014>
3	<kuid:-3:10049>
4	<kuid:44179:60021>
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Mesh

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- alastair.im** - The default mesh.
- alastair.texture.txt, alastair.tga** - The default mesh texture files.
- alastair_lowres.im** - A second mesh.
- alastair_lores.texture.txt, alastair_lores.tga** - The texture files for the second mesh.

File Listings

config.txt	
kuid	<kuid:56113:1003>
trainz-build	2.5
category-class	"HM"
category-region	"00"
category-era	"1980s;1990s;2000s"
username	"testMesh"
kind	"mesh"
description	"Sample Mesh Asset. This is a driver mesh."
mesh-table	
standing	
mesh	"alastair/Alastair.im"
auto-create	1

sitting	
mesh	"alastair_lowres/Alastair_lowres.im"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Mesh-Reducing-Track

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

brick.texture.txt, brick.tga, meshwall.jpg, wall.texture.txt, wall.tga - The texture files used by the indexed meshes for this asset.

default.im - The “short” mesh, which is the more detailed mesh which is used when the camera is close to the spline. This mesh is also used as the preview mesh in surveyor. Must be named “default.im”.

rockwall.im - The filename of the “long mesh”, which must be placed in a subdirectory of the same name as the mesh.

Only the file name is entered, not the directory name nor the file extension. For example, the full pathname and extension is “rockwall/rockwall.im”. Enter only “rockwall” in the text input box.

File Listings

config.txt	
bendy	0
carrate	0
casts_shadows	0

endlength	0
grounded	0.4
isroad	1
istrack	0
length	4
repeats	4
rgb	0,0,0
shadows	0
upright	0
visible-on-minimap	1
width	7.9
kuid	<kuid:56113:1008>
trainz-build	2.5
category-class	“TR”
username	“testMeshReducingTrack”
kind	“track”
unit_mesh	“rockwall”
category-region	“00”
category-era	“1980s;2000s;2010s”
description “Test Mesh Reducing Track asset. This asset appears in the ‘splines’ menu in surveyor. The asset has two distinct meshes, one for far and one for close. This illustrates the way in which Mesh-Reducing-Track works in-game.”	
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180

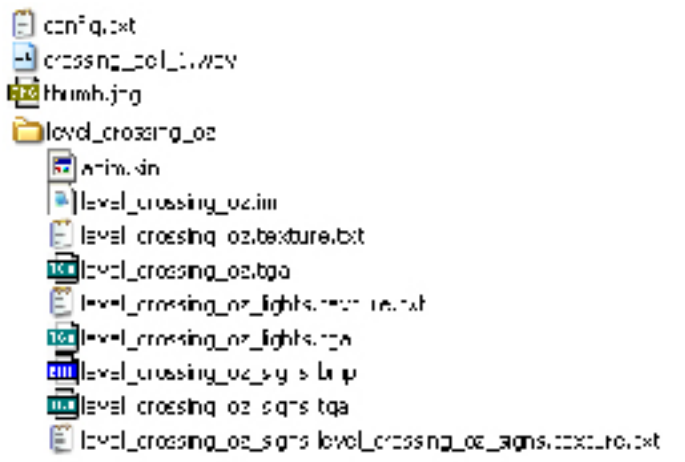
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

MOCrossing

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- crossing_bell_1.wav** - The environment sound of the crossing.
- anim.kin** - The animation file for the crossing.
- level_crossing_oz.im** - The default mesh of the crossing.
- level_crossing_oz.texture.txt, level_crossing_oz.tga** - Texture files for the default mesh of the crossing.
- level_crossing_oz_lights.texture.txt, level_crossing_oz_lights.tga** - The texture files for the signal lights.
- level_crossing_oz_signs.bmp, level_crossing_oz_signs.tga, level_crossing_oz_signs-level_crossing_oz_signs.texture.txt** - The texture files for the crossing signs.

File Listings

config.txt	
kuid	<kuid:56113:1261>
trainz-build	2.5
category-class	"WX"
category-region	"AU"
category-eras	"1970s;1980s;1990s;2000s;2010s"
username	"testMOCrossing"
kind	"mocrossing"
description	"An example MOCrossing Asset."
soundscript	
dayloop	
repeat-delay	0,0
distance	10,100
sound	
0	"crossing_bell_1.wav"
mesh-table	
default	
mesh	"level_crossing_oz/level_crossing_oz.im"
anim	"level_crossing_oz/anim.kin"
auto-create	1
attached-track	
road1	
track	<kuid:-1:100409>
useadjoiningtracktype	0
vertices	
0	"a.road0a"
1	"a.road0b"
track1	
track	<kuid:-1:100396>
useadjoiningtracktype	0
vertices	
0	"a.track0a"

1	"a.track0b"
track2	
track	<kuid:-1:100396>
useadjoiningtracktype	0
vertices	
0	"a.track1a"
1	"a.track1b"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

MOJunction

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

points.wav - The sound played when the junction is toggled.

grey.bmp, grey.texture.txt, orange.bmp, orange.texture.txt, rust.bmp, rust.texture.txt - The texture files for the Junction asset.

lever1.im, lever2.im - The indexed mesh files used for the junction levers.

File Listings

config.txt	
kuid	<kuid:56113:1254>
trainz-build	2.5
category-class	"WX"
category-region	"AG"
category-era	"1830s"
username	"testMOJunction"
kind	"mojunction"
light	1

trackside	0
description "Test MOJunction. Based on the "UK Point Motor" by Alan Thomson (snowsignal)."	
mesh-table	
lever1	
mesh	"lever1/lever1.im"
auto-create	1
lever2	
mesh	"lever2/lever2.im"
soundscript	
toggle	
trigger	"toggle"
distance	5,100
nostartdelay	1
repeat-delay	1,1
sound	
0	"points.wav"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

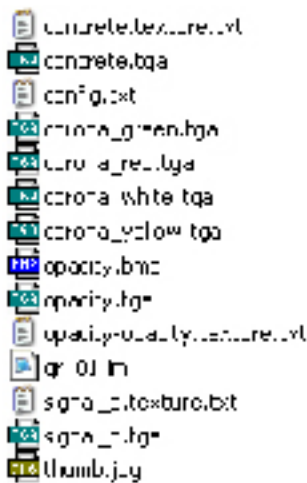
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

MOSignal

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- opacity.tga, opacity-opacity.texture.txt, concrete.texture.txt, concrete.tga, signal_q.tga** - The texture files used for the signal asset.
- corona_green.tga, corona_red.tga, corona_white.tga, corona_yellow.tga** - The corona textures used to light the signals.
- qr_01.im** - The signal indexed mesh file.

File Listings

config.txt	
kuid	<kuid:56113:1266>
trainz-build	2.5
category-class	“WA”
category-region	“AU”
category-era	“1980s;1990s;2000s;2010s”
username	“testMOSignal”
kind	“mosignal”
function	“TrackSignal”
description	“Sample MOSignal asset.”
trackside	-2.5
signals	

0	
light	11
2	
light	10,8,6,4,2,0
3	
light	10,8,7,5,3,1
4	
light	10
5	
light	9,8,6,4,2,0
6	
light	9,8,7,5,3,1
8	
light	9
9	
light	11,12,13
lights	
0	
corona	“corona_white.tga”
1	
corona	“corona_white.tga”
2	
corona	“corona_white.tga”
3	
corona	“corona_white.tga”
4	
corona	“corona_white.tga”
5	
corona	“corona_white.tga”
6	
corona	“corona_white.tga”
7	
corona	“corona_white.tga”
8	
corona	“corona_white.tga”

9	
corona	"corona_green.tga"
10	
corona	"corona_yellow.tga"
11	
corona	"corona_red.tga"
12	
corona	"corona_white.tga"
13	
corona	"corona_white.tga"
mesh-table	
default	
mesh	"QR_01.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

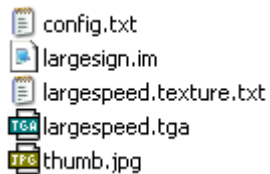
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

MOSpeedboard

Download this asset

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

largesign.im - The indexed mesh file used for the speedboard model.

largespeed.tga, largespeed.texture.txt - The texture files for the speedboard asset.

File Listings

config.txt	
kuid	<kuid:56113:1264>
trainz-build	2.5
category-class	"WS"
username	"testMOSpeedBoard"
kind	"mospeedboard"
category-region	"AU"
category-era	"2000s"
trackside	2.4
speedlimit	33.36
description	"Sample Speedboard Asset."
mesh-table	
default	
mesh	"LargeSign.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Pantograph

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

anim.kin - The animation file for the pantograph asset.

env_metal.bmp, env_metal.texture.txt, pant.texture.txt, pant.tga - The texture files for the pantograph asset.

panto.pm - The progressive mesh used for the pantograph.

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

File Listings

config.txt	
kuid	<kuid:56113:1003>
trainz-build	2.5
category-class	"ZP"
category-region	"00"
category-era	"1990s;2000s;2010s"
username	"testPantograph"
kind	"pantograph"
description	"Test pantograph asset."
thumbnails	
0	
width	240
height	180
image	"thumb.jpg"

Paintshed-Template

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

consist_mask.bmp - The template image used as a basis for the consist menu icon.

front_mask.bmp - The template image used as a basis for the texture on the front of the train model.

left_mask.bmp - The template image used as a basis for the texture on the left side of the train model.

main_mask.bmp - The template image used as a basis for the main texture of the traincar.

preview_mask.bmp -The template image used as a basis for the 512x512 preview image generated by paintshed.

rear_mask.bmp - The template image used as a basis for the texture on the rear of the train model.

right_mask.bmp - The template image used as a basis for the texture on the right side of the train model.

top_mask.bmp -The template image used as a basis for the texture on the top of the train model.

File Listings

config.txt	
kuid	<kuid:56113:1001>
trainz-build	2.5
category-class	"ZX"
category-region	"00"
category-era	"1970s;1980s;1990s;2000s;2010s"

username	"testPaintshed-Template"
kind	"paintshed-template"
paintshed-skin	<kuid:-13:132000>
description	"Test Paintshed-Template Asset."
thumbnails	0
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Paintshed-Skin

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

colormap.dat, skin.dat - Files containing extra color and skin information (generated by the paintshed program).

main.bmp, main.texture.txt - The texture files used for the actual train model in game.

alphanumeric_0a.tga, alphanumeric_0b.tga, alphanumeric_1a.tga, alphanumeric_1b.tga, alphanumeric_2a.tga, alphanumeric_2b.tga, alphanumeric_3a.tga, alphanumeric_3b.tga, alphanumeric_4a.tga, alphanumeric_4b.tga, alphanumeric_5a.tga, alphanumeric_5b.tga,

alphanumeric_6a.tga, alphanumeric_6b.tga, alphanumeric_7a.tga, alphanumeric_7b.tga, alphanumeric_8a.tga, alphanumeric_8b.tga, alphanumeric_9a.tga, alphanumeric_9b.tga - The running number images and opacity masks.

consist.bmp, consist_a.bmp, bb15000_art_icon.texture.txt - The image used as the train icon in the train menu. 128x64 bmp and accompanying alpha map.

preview.bmp, preview_a.bmp, bb15000_art_512.texture.txt - These image files are used to show a 512x512 preview texture when the image is available on the download station.

This is valid for 2004 assets, but in TRS2006 this functionality has been replaced with the 240x180 thumbnail image referenced in the thumbnails container.

If your asset has a trainz-version of 2.5+, you should use an image from a thumbnail container instead, at which time these files may be deleted if you wish to decrease the filesize of your asset.

File Listings

config.txt	
origin	"AU - AUSTRALIA"
category-class	"AA"
product-id	"paintshed"
product-version	1.4
product-type	"reskin"
pantograph	<kuid:-1:100860>
engine	1
interior	<kuid:-1:100554>
fonts	2
mass	90000
kind	"traincar"
enginespec	<kuid:-1:42004205>
enginesound	<kuid:-1:42003002>
hornsound	<kuid:-1:42003101>
username	"testPaintshed-Skin"
description	"paintshed skin. Generated in Paintshed and edited in CCP."
alias	<kuid:-10:182>
kuid	<kuid:56113:1005>
paintshed-template-used	<kuid:-13:157001>

paintshed-skin-used	<kuid:-13:157000>
category-region	"AU"
category-era	"2000s"
trainz-build	2.5
mesh-table	
default	
mesh bb15000_body.pm"	"bb15000_body/"
auto-create	1
shadow	
mesh bb15000_shadow.pm"	"bb15000_shadow/"
auto-create	0
bogeys	
0	
bogey	<kuid:-1:100005>
reversed	0
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

bb15000_art_icon.texture.txt

Primary=consist.bmp
 Tile=st
 Hint=Dynamic
 Alpha=consist_a.bmp

bb15000_art_512.texture.txt

Primary=preview.bmp
 Tile=st
 Hint=Dynamic
 Alpha=preview_a.bmp

Product (Coal Product)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- coal.tga, coal.texture.txt** - The texture file used when an item of rollingstock is carrying a load of coal. This texture is applied to the surface of the “load” in the rollingstock item, for example the load in the back of a coal hopper car.
- coal_icon.tga, icon_texture.texture.txt** - The product icon images (64x64 TGA).

File Listings

config.txt	
kuid	<kuid:56113:1004>
trainz-build	2.5
category-class	“IB”
category-region	“00;AE”
category-era	“1850s”
username	“testCoal”
kind	“product”
allows-mixing	1
instance-type	“resource”
mass	0.86
product-category	<kuid:-3:10040>
product-texture	“coal.texture”
thumbnails	
0	
width	240
height	180
image	“thumb.jpg”

1	
image	“Coal_icon.tga”
width	64
height	64

Download this asset

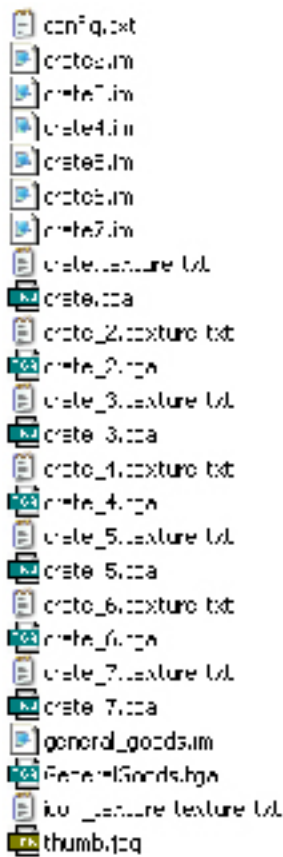
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Product (General Goods Product)

File Listings

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- general_goods.im, crate2.im, crate3.im, crate4.im, crate5.im, crate6.im, crate7.im** - The assorted indexed mesh files used to represent this product when placed on an item of rollingstock.
- Using more of these files creates a greater variety of meshes to be displayed when items of rollingstock are carrying this product.
- crate.tga, crate.texture.txt, crate_2.tga, crate_2.texture.txt, crate_3.tga, crate.texture_3.txt, crate_4.tga, crate_4.texture.txt, crate_5.tga, crate_5.texture.txt, crate_6.tga, crate_6.texture.txt, crate_7.tga, crate_7.texture.txt** - The texture files used by the indexed meshes of this product.
- generalgoods.tga, icon_texture.texture.txt** - The product icon images (64x64 TGA).

config.txt	
kuid	<kuid:56113:1005>
trainz-build	2.5
category-class	"IC"
category-region	"00"
category-era	"2000s"
username	"testGeneral Goods"
kind	"product"
allows-mixing	1
instance-type	"instance"
mass	1400
product-category	<kuid:-3:10042>
description	"Test General Goods Asset."
mesh-table	
default	
mesh	"general_goods.im"
auto-create	1
crate2	
mesh	"crate2.im"
auto-create	1
crate3	
mesh	"crate3.im"
auto-create	1
crate4	
mesh	"crate4.im"
auto-create	1
crate5	
mesh	"crate5.im"
auto-create	1
crate6	
mesh	"crate6.im"
auto-create	1
crate7	

mesh	"crate7.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"icon_texture.texture"
width	64
height	64

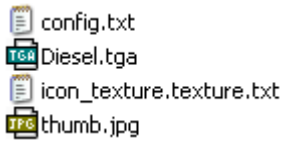
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Product (Diesel Fuel Product)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



height	64

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- icon_texture.texture.txt, diesel.tga** - The product icon images (64x64 TGA).

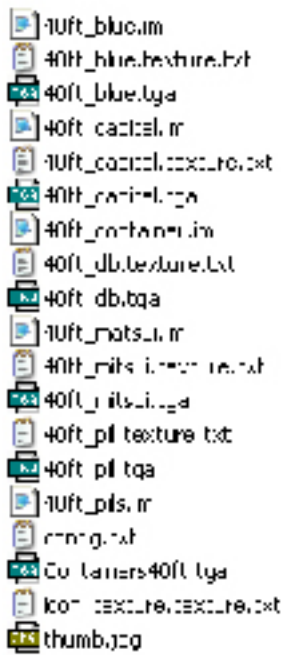
File Listings

config.txt	
kuid	<kuid:56113:1006>
trainz-build	2.5
category-class	“AA”
category-region	“00”
category-eras;2000s;2010s”	“1950s;1960s;1970s;1980s;1990s;2000s;2010s”
username	“testDiesel Fuel”
kind	“product”
allows-mixing	1
instance-type	“resource”
mass	0.89
product-category	<kuid:-3:10044>
description	“Treset Diesel Fuel product.”
thumbnails	
0	
image	“thumb.jpg”
width	240
height	180
1	
image	“icon_texture.texture”
width	64

Product (40ft Container Product)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- 40ft_blue.im, 40ft_capital.im, 40ft_container.im, 40ft_mitsui.im, 40ft_pils.im** - The assorted indexed mesh files used to represent this product when placed on an item of rollingstock.
- Using more of these files creates a greater variety of meshes to be displayed when items of rollingstock are carrying this product.
- 40ft_blue.texture.txt, 40ft_blue.tga, 40ft_capital.texture.txt, 40ft_capital.tga, 40ft_container.texture.txt, 40ft_container.tga, 40ft_db.texture.txt, 40ft_db.tga, 40ft_mitsui.texture.txt, 40ft_mitsui.tga, 40ft_pil.texture.txt, 40ft_pil.tga** - The texture files used by the indexed meshes of this product.
- Containers40ft.tga, icon_texture.texture.txt** - The product icon images (64x64 TGA).

File Listings

config.txt	
kuid	<kuid:56113:1007>
trainz-build	2.5
category-class	"IC"
username	"test40ft Container"

kind	"product"
allows-mixing	1
instance-type	"instance"
mass	22000
product-category	<kuid:-3:10042>
category-region	"00"
category-era	"1980s"
description	"Test 40ft Container asset."
mesh-table	
default	
mesh	"40ft_container.im"
auto-create	1
pils	
mesh	"40ft_pils.im"
auto-create	1
matsui	
mesh	"40ft_matsui.im"
auto-create	1
capital	
mesh	"40ft_capital.im"
auto-create	1
blue	
mesh	"40ft_blue.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"icon_texture.texture"
width	64
height	64

Product (Lumber Product)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- icon_texture.texture.txt, lumber.tga** - The product icon images (64x64 TGA).
- lumberstack.im** - The indexed mesh file used to represent this product when placed on an item of rollingstock.
- plank.texture.txt, plank.tga** - The texture files used by the indexed mesh of this product.

File Listings

config.txt	
kuid	<kuid:56113:1019>
trainz-build	2.5
category-class	"IB"
category-region	"00"
category-era	"1980s"
username	"testLumberProduct"
kind	"product"
allows-mixing	1
instance-type	"resource"
mass	8000
product-category	<kuid:-3:10042>
description	"Test Lumber Asset."
mesh-table	
default	
mesh	"lumberstack.im"
auto-create	1

thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"icon_texture.texture"
width	64
height	64

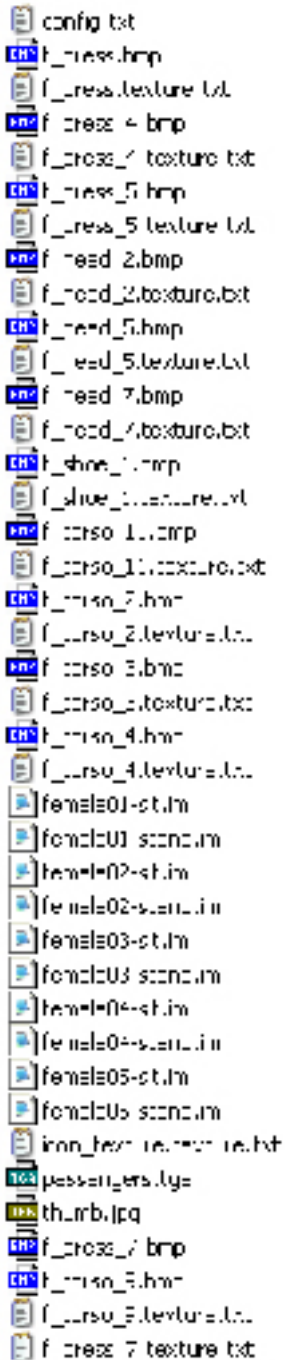
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Product (Passenger Product)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

f_dress.bmp, f_dress.texture.txt, f_dress_4.bmp,
f_dress_4.texture.txt, f_dress_5.bmp, f_dress_
5.texture.txt, f_head_2.bmp, f_head_2.texture.txt,
f_head_5.bmp, f_head_5.texture.txt, f_head_7.bmp,
f_head_7.texture.txt, f_shoe_1.bmp, f_shoe_1.texture.
txt, f_torso_11.bmp, f_torso_11.texture.txt, f_torso

2.bmp, f_torso_2.texture.txt, f_torso_3.bmp, f_torso_3.texture.txt, f_torso_4.bmp, f_torso_4.texture.txt, f_torso_9.texture.txt, f_torso_9.bmp, f_dress_7.bmp, f_dress7.texture.txt - The texture files used by the indexed meshes used in this product.

female01.sit.im, female01-stand.im, female02.sit.im, female02-stand.im, female03.sit.im, female03-stand.im, female04.sit.im, female04-stand.im, female05.sit.im, female05-stand.im - The assorted indexed mesh files used to represent passengers inside a populated railcar.

icon_texture.texture.txt, passengers.tga -The product icon images (64x64 TGA).

File Listings

config.txt	
kuid	<kuid:56113:1013>
trainz-build	2.5
category-class	"IP"
username	"testPassenger"
kind	"product"
allows-mixing	1
instance-type	"instance"
mass	65
product-category	<kuid:-3:10091>
category-region	"00"
category-era	"1990s;2000s;2010s"
description	"Test Passenger Product."
mesh-table	
female01-stand	
mesh	"Female01-Stand.IM"
female01-sit	
mesh	"Female01-Sit.IM"
female02-stand	
mesh	"Female02-Stand.IM"
female02-sit	
mesh	"Female02-Sit.IM"
female03-stand	
mesh	"Female03-Stand.IM"
female03-sit	

mesh	"Female03-Sit.IM"
female04-stand	
mesh	"Female04-Stand.IM"
female04-sit	
mesh	"Female04-Sit.IM"
female05-stand	
mesh	"Female05-Stand.IM"
female05-sit	
mesh	"Female05-Sit.IM"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"icon_texture.texture"
width	64
height	64
kuid-table	
0	<kuid:-3:10091>

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Product-Category

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

config.txt	
kuid	<kuid:56113:1250>
trainz-build	2.5
category-class	"IB"
category-region	"00"
category-era	"1970s;1980s;2000s;2010s"
username	"testProduct-Category"
kind	"product-category"
description	"Product Category. Useful when making traincar assets and designating which products are allowed,"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Profile

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

profile.dat - The data file containing information on the session. This file is generated by Trainz and is not human readable.

File Listings

config.txt

kind	"profile"
kuid	<kuid:56113:1245>
username	"testProfile"
map-kuid	<kuid:56113:1244>
category-class	"YS"
trainz-build	2.5
category-region	"AU"
category-era	"2000s"
description	"A Quick Test Profile. It's best to make these via Trainz and then edit them in CCP as desired."

kuid-table

0	<kuid:-16:10212>
1	<kuid:-16:2025>
2	<kuid:-101:10110>
3	<kuid:-3:10057>
4	<kuid:-3:10058>
5	<kuid:-3:10149>
6	<kuid:-3:10209>
7	<kuid:-3:10186>
8	<kuid:-3:10076>
9	<kuid:-3:10077>

10	<kuid:-3:10081>
11	<kuid:-3:10082>
12	<kuid:-3:10083>
13	<kuid:-3:10090>
14	<kuid:-3:10078>
15	<kuid:-1:1>
16	<kuid:56113:1244>

thumbnails

0	
image	"thumb.jpg"
width	240
height	180

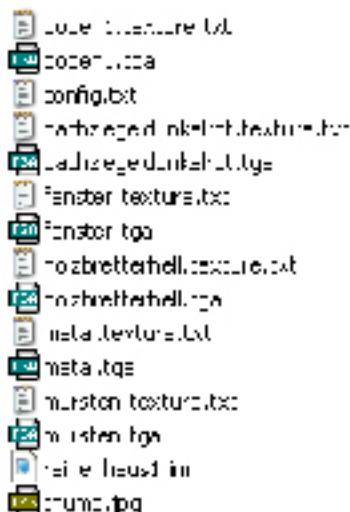
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Scenery

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

boden1.texture.txt, boden1.tga, dachziegweldunkelrot.texture.txt, dachziegweldunkelrot.tga, fenster.texture.txt, fenster.tga, holzbretterhell.texture.txt, holzbretterhell.tga, metal.texture.txt, metal.tga, mursten.texture.txt, mursten.tga - The texture files used for this scenery object.

reihenhaus1.im - The indexed mesh file used for this scenery object.

File Listings

config.txt	
kuid	<kuid:56113:1008>
trainz-build	2.5
category-class	"BH"
category-region	"00"
category-era	"1960s;1970s;1980s;1990s;2000s;2010s"
username	"testScenery"
kind	"scenery"
description	"Test Scenery object."
mesh-table	
default	

mesh	"reihenhaus1.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"
width	240

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Scenery-Trackside

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

-  catchpoint.im
-  config.txt
-  limit of shunt.texture.txt
-  limit of shunt.tga
-  thumb.jpg

Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- catchpoint.im** - The indexed mesh of this scenery-trackside asset.
- limit of shunt.texture.txt, limit of shunt.tga** - The texture files used by the indexed mesh of this asset.

width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

File Listings

config.txt	
kuid	<kuid:56113:1009>
trainz-build	2.5
category-class	"WS"
category-region	"00"
category-era	"1860s"
username	"testTrackside"
kind	"scenery"
trackside	-2.8
description	"Test Trackside object."
mesh-table	
default	
mesh	"CatchPoint.im"
auto-create	1
thumbnails	
0	
image	"thumb.jpg"

Steam-Engine

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

config.txt	
kuid	<kuid:56113:1235>
trainz-build	2.5
category-class	“ZE”
category-region	“US”
category-era	“1940s”
username	“testSteamEngine”
kind	“steam-engine”
description	“Test Steam Engine asset. Based on the UP Big Boy engine file.”
flowsize	
trainbrakepipe	170000
epreservoirpipe	0.1
no3pipe	0.1
no4pipe	0.1
auxreservoirvent	0.1
auxreservoir_no3	0.1
auxreservoir_trainbrakepipe	0.1
autobrakecylindervent	0.1
auxreservoir_autobrakecylinder	0.1
equaliser_mainreservoir	0.06
equaliservent	0.06
equaliserventhandleoff	0.1
equaliserventemergency	0.1

no3pipeevent	1.5
no3pipe_mainreservoir	0.1
compressor	5
trainbrakepipe_reservoir	1
trainbrakepipevent	0.06
no3pipe_autobrakecylinder	0.1
epreservoirpipe_autobrakecylinder	0.1
mainreservoir_ep	0.1
vacuumbrakepipe	0.1
vacuumbrakepipereleasevent	0.1
vacuumbrakepipevent	0.1
vacuumbrakereservoir_vacuumbrakepipe	0.1
vacuumbrakecylinder_vacuumbrakepipe	0.1
highspeedexhauster_vacuumbrakepipe	0.1
volume	
scale	1
trainbrakepipe	0.2
epreservoirpipe	0.2
no3pipe	0.2
no4pipe	0.2
auxreservoir	0.0384678
autobrakecylinder	0.00969387
vacuumbrakepipe	0
vacuumbrakereservoir	0
vacuumbrakecylinder	0
mainreservoir	1
equaliser	0.5
independantbrakecylinder	0.0103239
pressure	
scale	1
compressor	0.00946941
mainreservoir	0.00946941
highspeedexhauster	0
brakepipe	0.00595441
brakeinitial	0.00560291

brakefull	0.00398601	maximum-volume	81000
indbrakefull	0.00398601	initial-boiler-temperature	455
trainbrakepipe_start	0.00440781	water-injector-rate	20
epreservoirpipe_start	0	piston-volume-min	8.715
no3pipe_start	0	piston-volume-max	232.4
no4pipe_start	0	piston-area	0.285
auxreservoir_start	0.00504051	piston-angular-offsets	0.0174,0.8028,1.5254,2.3736,3.0333,3.9444,4.5413,5.5152
autobrakecylinder_start	0.00489991	firebox-to-boiler-heat-flow-idle	0.003
vacuumbrakepipe_start	0	burn-rate-idle	0.003
vacuumbrakereservoir_start	0	boiler-to-piston-flow	0.0039
vacuumbrakecylinder_start	0	piston-to-atmosphere-flow	0.0031
mainreservoir_start	0.00876641	safety-valve-low-pressure	2164.97
equaliser_start	0.00440781	safety-valve-low-flow	0.00375
independantbrakecylinder_start	0.00489991	safety-valve-high-pressure	2179
mass		safety-valve-high-flow	0.003875
scale	1	max-fire-coal-mass	585
fuel	"6.2156e+006"	max-fire-temperature	1585
motor		shovel-coal-mass	55
resistance	1.75	burn-rate	0.0825
adhesion	4.7	fuel-energy	27.5
maxvoltage	600	firebox-volume	0
maxspeed	30	main-reservoir-volume	0
brakeratio	66000	westinghouse-volume	0
max-accel	5200	thumbnails	
max-decel	85000	0	
axle-count	16	image	"thumb.jpg"
surface-area	100	width	240
moving-friction-coefficient	0.035	height	180
air-drag-coefficient	0.002		
throttle-notches	8		
steam			
firebox-to-boiler-heat-flow	0.07125		
firebox-efficiency	0.9		
boiler-volume	95000		
minimum-volume	73000		

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Texture

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

 config.txt
 eg.bmp
 thumb.jpg

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

eg.bmp - The image file used as the texture in this asset.

File Listings

config.txt	
kuid	<kuid:56113:1267>
trainz-build	2.5
category-class	"JO"
category-region	"AU"
category-era	"2000s;2010s"
username	"testTexture"
kind	"texture"
texture	"eg.bmp"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Texture-Group

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

1.bmp, 2.bmp, 3.bmp - The texture files stored within this asset (which will be referenced via script from another asset).

File Listings

config.txt	
kuid	<kuid:56113:1002>
trainz-build	2.5
category-class	"JO"
category-region	"00"
category-era	"1810s"
username	"testTextureGroup"
kind	"texture-group"
description	"Test texture group."
textures	
0	"1.bmp"
1	"2.bmp"
2	"3.bmp"
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

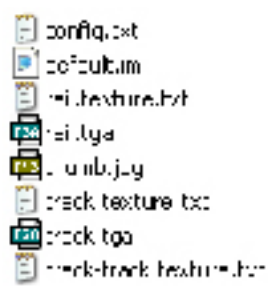
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Track

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.
- default.im** - The indexed mesh used for the track asset. Should be named "default.im".
- rail.texture.txt, rail.tga, track.texture.txt, track.tga, track-track.texture.txt** - The texture files used by this track asset.

File Listings

config.txt	
bendy	1
carrate	0
casts_shadows	0
endlength	40
grounded	0.4
isroad	0
istrack	1
length	2
repeats	1
rgb	255,200,0
shadows	0
upright	0
visible-on-minimap	1
width	4
kuid	<kuid:56113:1006>
trainz-build	2.5

category-class	"TR"
category-region	"00"
category-era	"2000s"
username	"testTrack"
kind	"track"
description	"Sample Track."
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

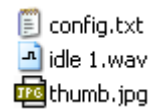
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Tracksound

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

idle 1.wav - The track sound file.

File Listings

config.txt	
kuid	<kuid:56113:1001>
trainz-build	2.5
category-class	"XSN"
category-region	"00"
category-era	"1930s;1940s;1950s;1960s;1970s;1980s;1990s;2000s;2010s"
username	"testTracksound"
kind	"tracksound"
min-distance	10
max-distance	10000
description	"Sample Tracksound asset."
levels	
0	0.1
1	10
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

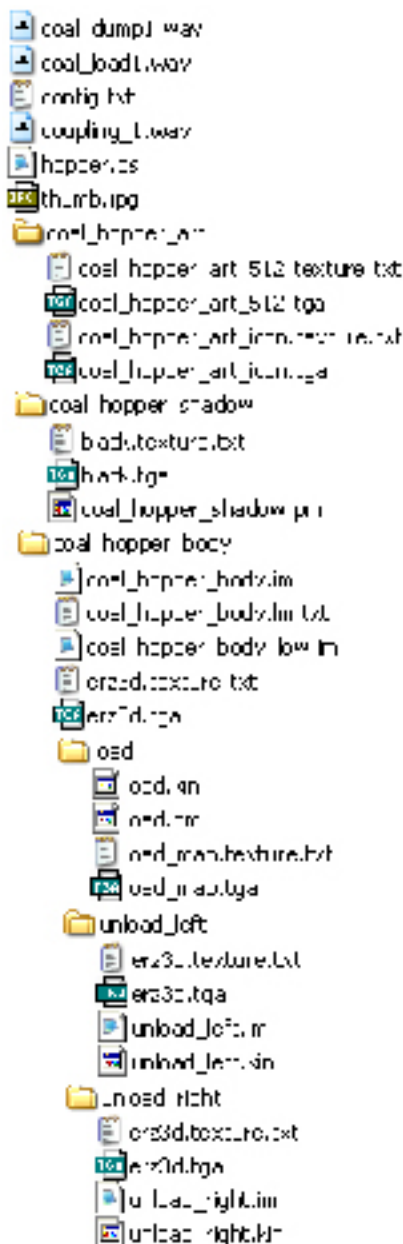
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Traincar (Coal Hopper)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

coal_dump1.wav, coal_load1.wav, coupling_1.wav - Sound files referenced within the hopper.gs script file.

hopper.gs - Script file detailing some functionality of the traincar asset.

coal_hopper_art_512.texture.txt, coal_hopper_art_512.tga - These image files are used to show a 512x512 preview texture when the image is available on the download station.

This is valid for 2004 assets, but in TRS2006 this functionality has been replaced with the 240x180 thumbnail image referenced in the thumbnails container.

If your asset has a trainz-version of 2.5+, you should use an image from a thumbnail container instead, at which time these files may be deleted if you wish to decrease the filesize of your asset.

coal_hopper_art_icon.texture.txt, coal_hopper_art_icon.tga - The icon texture files. 128x64 pixels in dimension.

coal_hopper_body.lm.txt - Level of Detail (or 'LOD') file. See the section on LOD meshes in Chapter 8 for more information.

erz3d.texture.txt, erz3d.tga, black.texture.txt, black.tga, load_map.texture.txt, load_map.tga - The texture files used by the indexed and progressive meshes.

load.pm, coal_hopper_shadow.pm - The progressive meshes used by the traincar asset.

unload_left.kin, unload_right.kin, load.kin - The animation files used by the traincar asset for loading and unloading operations.

unload_left.im, unload_right.im, coal_hopper_body.im, coal_hopper_body_low.im - The indexed mesh files used by the traincar asset.

File Listings

config.txt	
kuid	<kuid:56113:1001>
trainz-build	2.5
category-class	"XG"
username	"testTraincar (Hopper)"
kind	"traincar"
engine	0
mass	15000
category-region	"00"
category-era	"1990s;2000s;2010s"
enginespec	<kuid:-1:42004201>
script	"hopper.gs"
class	"Hopper"
icon0	<kuid:-3:10164>
description	"Test Coal Hopper Asset."
soundscript	
door_close	

trigger	"door_close"
nostartdelay	1
repeat-delay	1,0
distance	5,170
sound	
0	"coupling_1.wav"
mesh-table	
default	
mesh hopper_body.lm"	"coal_hopper_body/coal_
auto-create	1
shadow	
mesh hopper_shadow.pm"	"coal_hopper_shadow/coal_
load	
mesh pm"	"coal_hopper_body/load/load.
anim kin"	"coal_hopper_body/load/load.
auto-create	1
use-parent-bounds	1
effects	
product-texture	
kind	"texture-replacement"
texture	"load_map.texture"
left-door	
mesh unload_left.im"	"coal_hopper_body/unload_left/
anim unload_left.kin"	"coal_hopper_body/unload_left/
auto-create	1
right-door	
mesh unload_right.im"	"coal_hopper_body/unload_right/
anim unload_right.kin"	"coal_hopper_body/unload_right/
auto-create	1
queues	

load0	
size	54300
initial-count	0
animated-mesh	"load"
product-kuid	<kuid:44179:60013>
allowed-categories	
0	<kuid:-3:10040>
smoke0	
attachment	"a.unload_left_pfx0"
mode	"time"
color	0,0,0,250
rate	8
velocity	2
lifetime	2
minsize	1
maxsize	4
enabled	0
smoke1	
attachment	"a.unload_left_pfx1"
mode	"time"
color	0,0,0,250
rate	8
velocity	2
lifetime	2
minsize	1
maxsize	4
enabled	0
smoke2	
attachment	"a.unload_right_pfx0"
mode	"time"
color	0,0,0,250
rate	8
velocity	2
lifetime	2
minsize	1

maxsize	4
enabled	0
smoke3	
attachment	"a.unload_right_pfx1"
mode	"time"
color	0,0,0,250
rate	8
velocity	2
lifetime	2
minsize	1
maxsize	4
enabled	0
bogeys	
0	
bogey	<kuid:-1:100063>
reversed	0
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180
1	
image	"coal_hopper_art/coal_hopper_
art_icon.texture"	
width	128
height	64

```

hopper.gs

//

// Hopper.gs

//

// Copyright (C) 2003 Auran Developments Pty Ltd

// All Rights Reserved.

//

//

include "vehicle.gs"

//

// Hopper scriptlet class. No threads - only call overridden
callbacks from Vehicle that

// are called by an Industry.

//

class Hopper isclass Vehicle
{

    // Play sound hopper is starting to be loaded.

    float BeginLoad(LoadReport report)
    {
        Asset meAsset = GetAsset();

        World.PlaySound(meAsset, "coal_load1.wav", 1000.0f,
20.0f, 1000.0f, me, "");

        return 0.0;
    }

    // Activate particles and play a sound as the hopper is
beginning to unload.

    float BeginUnload(LoadReport report)
    {
        SetMeshAnimationState("left-door", true);

        SendMessage(me, "pfx", "+0+1");

        SetMeshAnimationState("right-door", true);

        SendMessage(me, "pfx", "+2+3");

        Asset meAsset = GetAsset();

        World.PlaySound(meAsset, "coal_dump1.wav", 1000.0f,
20.0f, 1000.0f, me, "");

        return 1.0;
    }
}

```

```

}

// Deactivate particles and play a sound as the hopper is
ending the unload operation.

float EndUnload>LoadingReport report)
{
    Sleep(1.0);

    SetMeshAnimationState("left-door", false);

    SendMessage(me, "pfx", "-0-1");

    SetMeshAnimationState("right-door", false);

    SendMessage(me, "pfx", "-2-3");

    return 1.0;
}
};

```

coal_hopper_body.lm.txt

```

version 1.0

offset = 0.01;

calcPoint = center;

multiplier = 1.0;

animationCutOff = 0.00;

mesh("0.25")
{
    name="coal_hopper_body_low.im";
}

mesh("1.0")
{
    name="coal_hopper_body.im";
}

```

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Traincar (Diesel Engine)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

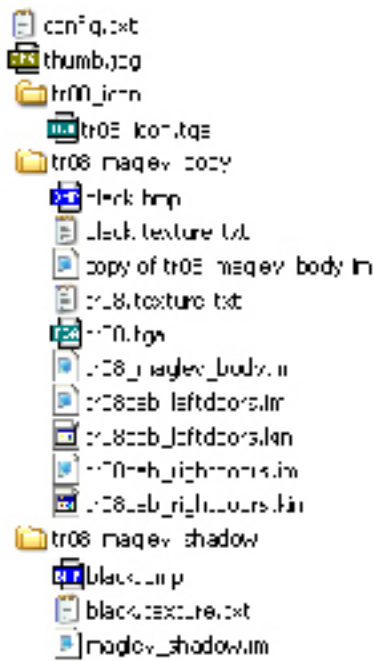
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Traincar (Electric Engine)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

- config.txt** - The config file for the asset.
- thumb.jpg** - The thumbnail image for this asset. A 240x180 jpeg.

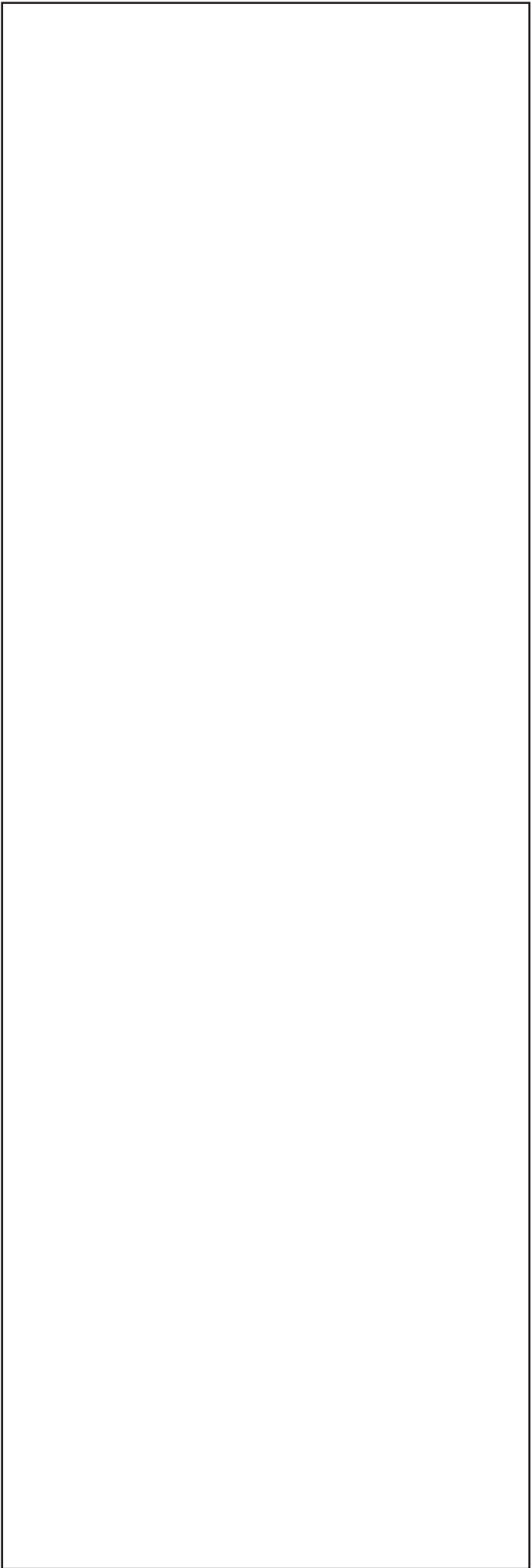
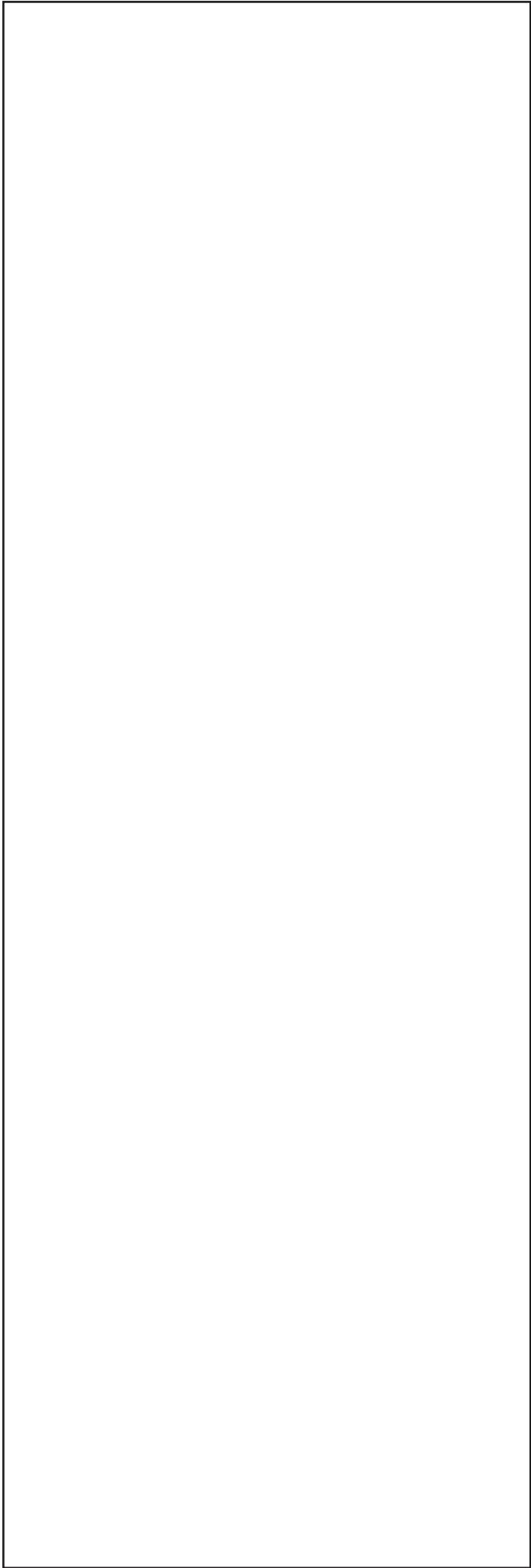
File Listings

config.txt	
kuid	<kuid:56113:1003>

trainz-build	2.5
category-class	"AE"
username	"testTrainCar (Electric)"
kind	"traincar"
engine	1
mass	53000
company	"Transrapid International"
origin	"Germany"
disable-extra-track-sounds	1
enginespec	<kuid:37522:2>
enginesound	<kuid:-1:42003002>
hornsound	<kuid:60723:54000>
category-region	"DE"
category-era	"2000s"
fonts	0
max-coupler-gap	0
use-coupler-sounds	0
description	
asset. Based on the Maglev."	
mesh-table	
default	
mesh	"TR08_Maglev_body/
TR08_Maglev_body.im"	
auto-create	1
shadow	
mesh	"TR08_Maglev_shadow/
Maglev_shadow.im"	
left-passenger-door	
mesh	"TR08_Maglev_body/
TR08Cab_LeftDoors.im"	
anim	"TR08_Maglev_body/
TR08Cab_LeftDoors.kin"	
auto-create	1
att	"a.doors"
att-parent	"default"
right-passenger-door	

mesh	"TR08_Maglev_body/TR08Cab_RightDoors.im"
anim	"TR08_Maglev_body/TR08Cab_RightDoors.kin"
auto-create	1
att	"a.doors"
att-parent	"default"
queues	
passengers	
passenger-queue	"1"
product-kuid	<kuid:-3:10060>
size	92
initial-count	5
bogeys	
bogey-element0	
reversed	0
bogey	<kuid2:171456:149:1>
thumbnails	
0	
image	"tr08_icon/tr08_icon.tga"
width	128
height	64
1	
image	"thumb.jpg"
width	240
height	180
kuid-table	
0	<kuid:-1:100141>
1	<kuid:37522:2>
2	<kuid:-1:42003002>
3	<kuid:60723:54000>
4	<kuid:-10:216>
5	<kuid:-3:10060>
6	<kuid2:171456:149:1>

Download
This as
website
theURL



Traincar (Rollingstock)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Traincar (Passenger Car)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

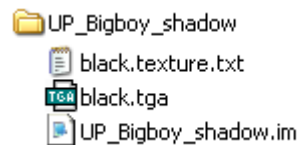
Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Traincar (Steam Locomotive)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

bigboy-cyl-steam.tfx, bigboy-smoke.tfx, bigboy-smoke-idle.tfx, bigboy-steam-relief.tfx - The Twinkle files used for the smoke effects. [More information on twinkles can be found in Chapter 11.](#)

UP_Bigboy_art_512.texture.txt, UP_Bigboy_art_512.tga - These image files are used to show a 512x512 preview texture when the image is available on the download station.

This is valid for 2004 assets, but in TRS2006 this functionality has been replaced with the 240x180 thumbnail image referenced in the thumbnails container.

If your asset has a trainz-version of 2.5+, you should use an image from a thumbnail container instead, at which time these files may be deleted if you wish to decrease the filesize of your asset.

UP_Bigboy_art_icon.texture.txt, UP_Bigboy_art_icon.tga - The icon texture files. 128x64 pixels in dimension.

bb_alpha.bmp, bb_alpha.texture.txt, bb_alpha.tga, bb_alpha-bb_alpha.texture.txt, BB_Bolts.texture.txt, BB_Bolts.tga, BB_Bolts_bump.texture.txt, BB_Bolts_bump.tga, BB_Bolts_LOW.texture.txt, BB_Bolts_LOW.tga, BB_Bolts_med.texture.txt, BB_Bolts_med.tga, bb_main01.texture.txt, bb_main01.tga, bb_main.texture.txt, bb_main.tga, bb_main_bump.texture.txt, bb_main_bump.tga, bb_main_LOW.texture.txt, bb_main_LOW.tga, bb_main_MED.texture.txt, bb_main_MED.tga, Env_glass.bmp, Env_glass.texture.txt, Env_metal.bmp, Env_metal.texture.txt, generic.texture.txt, generic.tga, generic_LOW.texture.txt, generic_LOW.tga, generic_med.texture.txt, generic_med.tga, window+interior.texture.txt, window+interior.tga, black.texture.txt, black.tga - The texture files used by the indexed meshes.

UP_Bigboy_body.im.txt - Level of Detail (or 'LOD') file. [See the section on LOD meshes in Chapter 8 for more information.](#)

UP_Bigboy_body.im, UP_Bigboy_body_low.im, UP_Bigboy_body_med.im, UP_Bigboy_shadow.im - The indexed mesh files used by the traincar asset.

File Listings

config.txt			
kuid	<kuid:56113:1002>	rate	1
trainz-build	2.5	velocity	0.8
category-class	"AS"	lifetime	0.4
username	"testTraincar (Steam)"	minsize	0.5
kind	"traincar"	maxsize	1.5
engine	1	smoke1	
mass	544310	attachment	"a.steam_cyl_drainR"
category-region	"US"	mode	"anim"
category-era s;1980s"	"1930s;1940s;1950s;1960s;1970	color	255,255,255,225
enginespec	<kuid:523:51469>	start	0.1
enginesound	<kuid:-3:10105>	period	0
hornsound	<kuid:523:54745>	rate	1
smoke_fastlife	6	velocity	0.8
smoke_fastspeed	2	lifetime	0.4
smoke_height	0	minsize	0.5
smoke_random	2	maxsize	1.5
smoke_shade	0.3	smoke2	
smoke_slowlife	1	attachment	"a.steam_L0"
description	"Test steam traincar asset. Based on the UP BigBoy."	mode	"anim"
mesh-table		color	255,255,255,225
default		start	0.61
mesh	"UP_Bigboy_body/UP_Bigboy_	period	0
body.im"		rate	1
auto-create	1	velocity	0.8
shadow		lifetime	0.4
mesh	"UP_Bigboy_shadow/UP_	minsize	0.5
Bigboy_shadow.im"		maxsize	1.5
smoke0		smoke3	
attachment	"a.steam_cyl_drainL"	attachment	"a.steam_R0"
mode	"anim"	mode	"anim"
color	255,255,255,225	color	255,255,255,225
start	0.6	start	0.11
period	0	period	0
		rate	1
		velocity	0.8

lifetime	0.4
minsize	0.5
maxsize	1.5
smoke4	
attachment	"a.safety01"
mode	"time"
color	255,255,255,150
rate	45
velocity	0.5
lifetime	0.4
minsize	0.05
maxsize	0.5
smoke5	
attachment	"a.safety02"
mode	"time"
color	255,255,255,150
rate	45
velocity	0.5
lifetime	0.4
minsize	0.05
maxsize	0.5
smoke6	
attachment	"a.smoke0"
mode	"speed"
color	50,50,50,255
accel	0,0,-1.5
start	0,5,10,20
rate	5,12,15,20
velocity	2.5,3.5,4.5,5.5
lifetime	2,3,4,4
minsize	0.4
maxsize	2,3,4,5
smoke7	
attachment	"a.smoke1"
mode	"speed"

color	50,50,50,255
accel	0,0,-1.5
start	0,5,10,20
rate	5,12,15,20
velocity	2.5,3.5,4.5,5.5
lifetime	2,3,4,4
minsize	0.4
maxsize	2,3,4,5
bogeys	
0	
bogey	<kuid:523:10071>
reversed	0
1	
bogey	<kuid:523:10072>
reversed	0
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

UP_Bigboy_body.lm.txt

```
version 1.0

offset = 0.01;

calcPoint = center;

multiplier = 1.0;

animationCutOff = 0.00;

renderCutOff = 0.00;

attachmentCutOff = 0.06;

mesh("0.25")

{
    name="UP_Bigboy_body_low.im";
}

mesh("0.52")

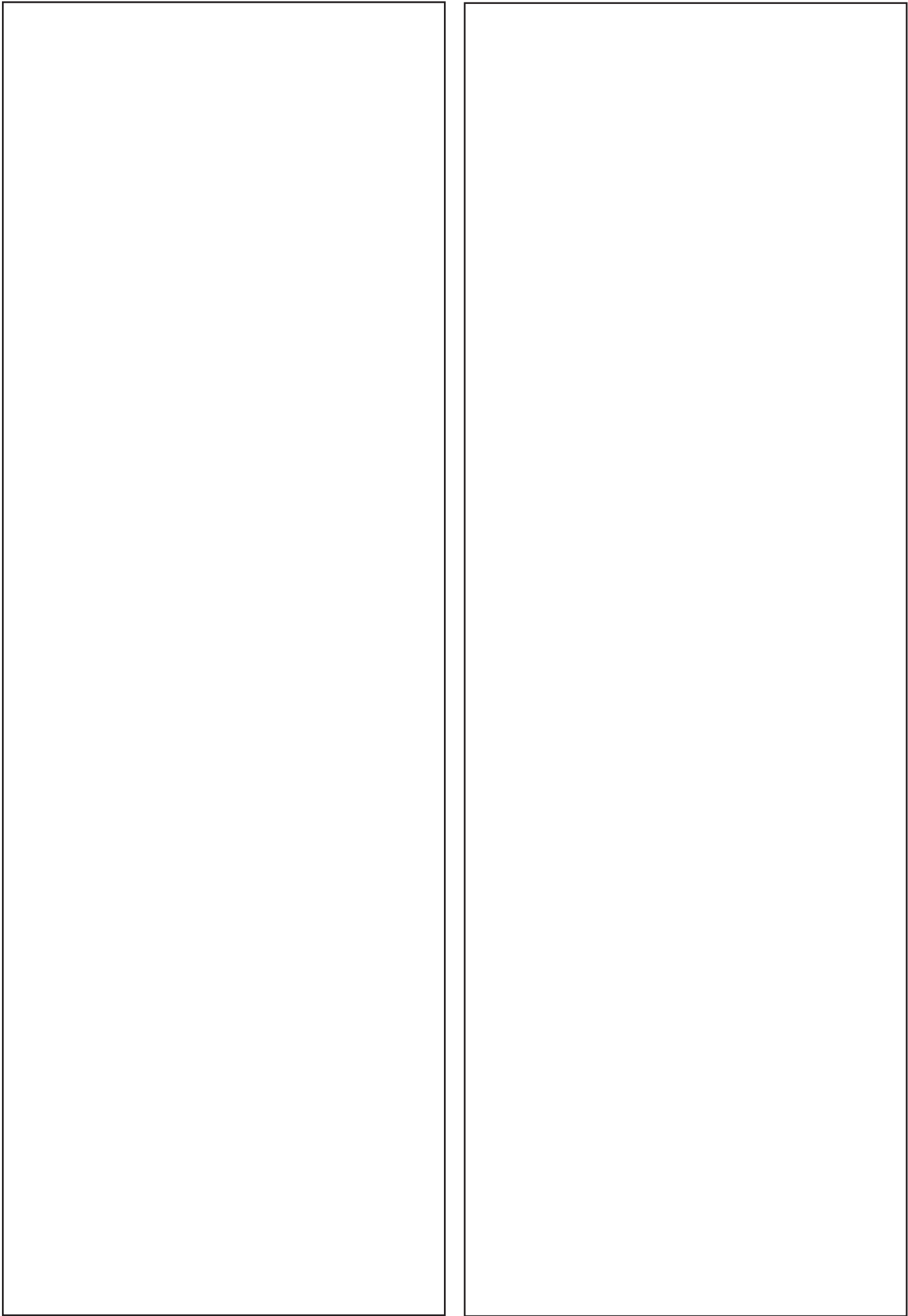
{
    name="UP_Bigboy_body_med.im";
}

mesh("1.0")

{
    name="UP_Bigboy_body.im";
}
```

Download this asset

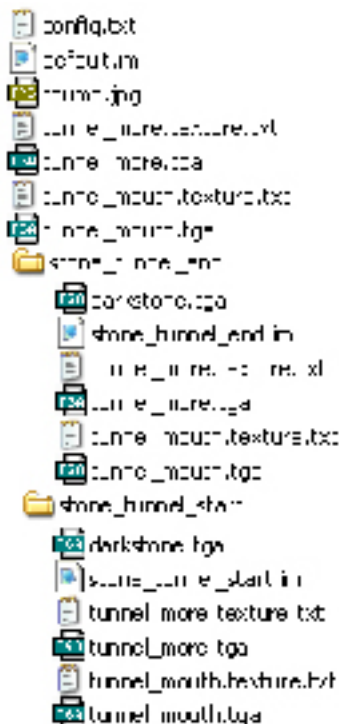
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Tunnel

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

default.im - The middle section of the tunnel asset. This mesh is also used as the preview image. Must be named "default.im" and placed in the base directory.

stone_tunnel_end.im, stone_tunnel_start.im - The indexed meshes used at each end of the tunnel.

tunnel_more.texture.txt, tunnel.re.tga, tunnel_mouth.texture.txt, tunnel_mouth.tga, darkstone.tga, tunnel_more.texture.txt, tunnel_more.tga - The texture files used by the indexed meshes for this asset.

File Listings

config.txt	
bendy	0
carrate	55
casts_shadows	0
endlength	20
grounded	0
isroad	0

istrack	1
length	20
repeats	0
rgb	180,180,180
shadows	0
upright	0
visible-on-minimap	1
width	7.9
kuid	<kuid:56113:1006>
trainz-build	2.5
category-class	"TT"
category-region	"AG"
category-era	"1830s"
username	"testTunnel"
kind	"bridge"
bridgetrack	<kuid:-1:100608>
height	12
trackoffsets	-2,2
initiator	"stone_tunnel_start"
terminator	"stone_tunnel_end"
description	
"Test Tunnel asset."	
thumbnails	
0	
image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Turntable (Animated)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:

Required Files

config.txt - The config file for the asset.

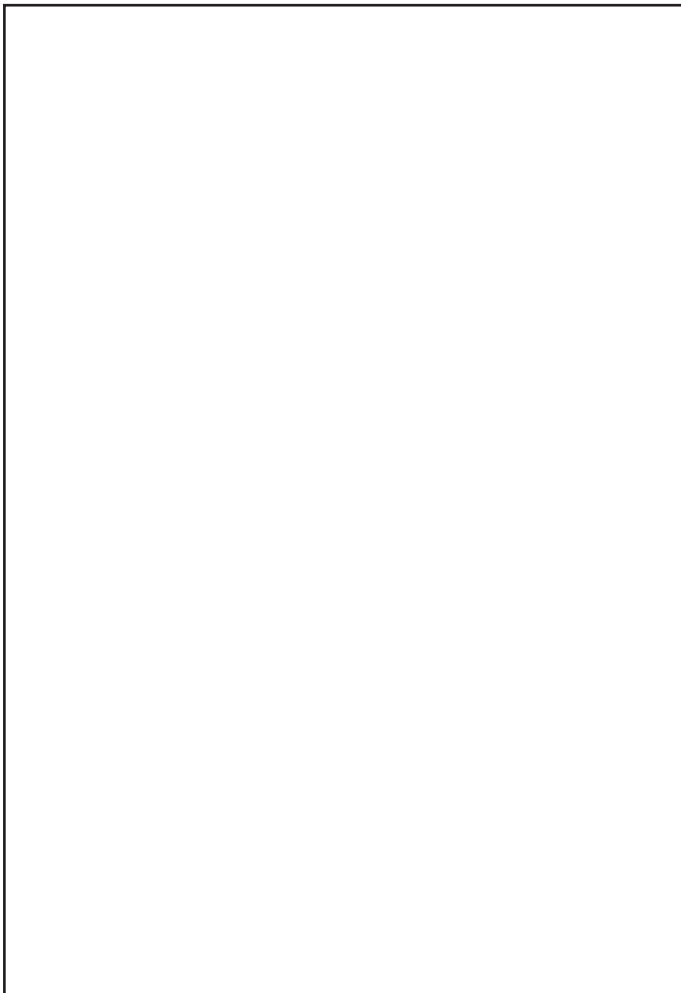
thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings



Download this asset

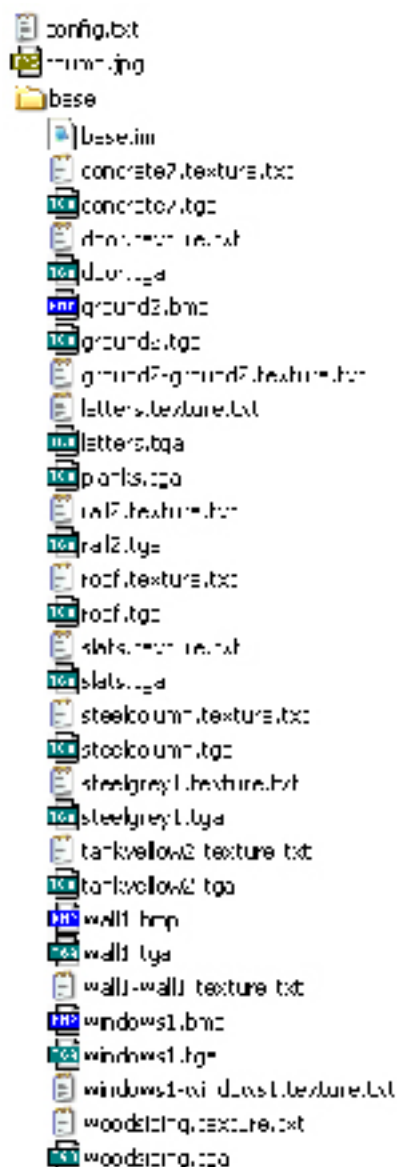
This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.



Turntable (Not animated)

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

File Listings

config.txt

kuid <kuid:56113:1011>

kind "turntable"

username "testTurntable"

trainz-build 2.5

category-class "BR"

category-region "00"

category-era "1850s;1860s;1870s;1880s"

snapmode 1

dighole 4,4

light 1

angle 0,10,20,30,40,50,60,180,190,200,210,220,230,240

looping 0

description "Test turntable asset."

mesh-table

default

mesh "base/base.im"

auto-create 1

turntable

mesh "turntable/turntable.im"

auto-create 1

attached-track

track_turntable

track <kuid:11:32001>

useadjoiningtracktype 0

vertices

0 "a.itrack0a"

1 "a.itrack0b"

track0_base

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack0a"

1 "a.otrack0b"

track1_base

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack1a"

1 "a.otrack1b"

track2_base

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack2a"

1 "a.otrack2b"

track3_base_

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack3a"

1 "a.otrack3b"

track4_base

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack4a"

1 "a.otrack4b"

track10_base

track <kuid:9:50001>

useadjoiningtracktype 0

vertices

0 "a.otrack10a"

1 "a.otrack10b"

kuid-table

0 <kuid:9:50001>

1 <kuid:11:32001>

thumbnails

0

image	"thumb.jpg"
width	240
height	180

Download this asset

This asset is available for download from the TRS2006 website at: <http://www.auran.com/TRS2006/downloads/theURLhere.zip>.

Water2

Directory Structure

A typical asset of this kind has the following File\Directory Structure:



Required Files

config.txt - The config file for the asset.

thumb.jpg - The thumbnail image for this asset. A 240x180 jpeg.

water.anim.txt - Contains the animation variables for the water asset.

water.bmp, **wateropacity.texture.txt** - The texture files applied to the water.

water.config.txt - Contains the tile size and material properties for the water asset.

File Listings

config.txt

kuid	<kuid:56113:1226>
trainz-build	2.5
category-class	"EW"
category-region	"00"
category-era	"2010s"
username	"TestWater"
kind	"water2"
description	"This is a test water2 kind written for the 2006 CCG."

thumbnails

0	
image	"thumb.jpg"
width	240
height	180

water.anim.txt

```
version 1.00

// Water DetailAnim configuration file

// Is used from DefaultWater.config.txt

DetailAnim
{
    AnimName = WaterAnimationPN;      // Perlin noise
    animation.

    AnimSampleRate = 10;               // Sample rate (samples per
    sec)

    AnimPeriod = 5;                   // Looping period in sec.

    AnimSpeed = 5.0;                  // Speed of waves

    AnimSize = 128,128;               // Bump map dimentions
    //AnimSize = 32,32;               // Bump map dimentions

    AnimWorldSize = 450.0;            // "Size" of one tile

    AnimMaxHeight = 1.0;              // Max height of the wave

    AnimScaleNormXY = 4.0;            // scale X,Y coordinates
    of the normal map for better interpolation

    Perlin
    {
        AnimPerlinFreq = 15.0;

        AnimPerlinSpeed = 10.0;

        AnimPerlinOctaves = 3.0;
    }

    FFT
    {
        AnimFFTWindVec = -15.0,5.0;   // Direction and speed
        of the wind affecting length of the waves

        AnimPhillipsA = 1.0e-3;       // Phillips spectrum constant
        affecting heights of the waves

        AnimFFTSeed = 0;

        // Reduce height of waves in direction perpendicular to wind
        (CosMin <= cos(dir) <= CosMax)

        //acrossCosMax    // Default -1.1 (disabled) (1.0 same
        direction, 0.0 - perpendicular, -1.0 opposite)

        //acrossCosMim    // Default 1.1

        //acrossRatio     // Default 1.0
```

```
// Reduce height of waves in direction opposite to wind

//oppositeCos    // Default -1.1 (disabled) (1.0 same
direction, -1.0 opposite direction)

//oppositeRatio  // Default 1.0

}

}
```

water.config.txt

version 1.00

// WaterManager config data

WaterManager("WaterManagerGeneric")

{

WaterMaterial

{

materialColor = (0.20, 0.45, 0.45, 0.8);

materialRI = 0.3;

opacityTex = WaterOpacity.texture;

opacityAmount = 0.5;

}

// Compiled DetailAnim or text ConfigData file 'water.anim.
txt'

// This is now loaded manually by Trainz so Trainz can cache
the anim file in a separate folder.

// DetailAnimFile = water.anim;

}

// WaterGeometry config data

WaterGeometry

{

UVScrollVelocity = 0.0, 0.05;

TileUVScale = 1.0, 1.0;

GridSpacing = 10.0; // "Size" of one cell of the
grid (is used if MaxAmp > 0)

TileGridSize = 2, 2; // Number of vertices in one
tile (use more if MaxAmp > 0)

WaveFreq = 0.0; // 0.15;

MaxAmp = 0.0; // 0.25;

// Mesh animation

// TileGridSize = 3, 3; // Number of vertices in
one tile (use more if MaxAmp > 0)

// WaveFreq = 0.15;

// MaxAmp = 0.25;

}

Download this asset

This asset is available for download from the TRS2006
website at: [http://www.auran.com/TRS2006/downloads/
theURLhere.zip](http://www.auran.com/TRS2006/downloads/theURLhere.zip).

GROUNDTEXTURE

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Ground\

Typical directory structure for a custom ground texture should be:



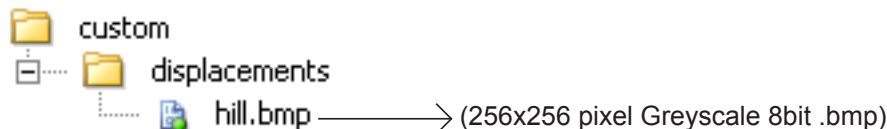
DISPLACEMENTS

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\displacements\

Displacement maps are used to adjust the height and shape of an area of terrain.

Typical directory structure for a custom ground texture should be:



Note: Displacement maps do not require a config.txt file.

ENVIRONMENT

The Environment folder accommodates the use of different types of sky and water in TRS.

WATER2

New animated water in TRS (kind water2) supercedes previous UTC water (kind water). Refer to descriptions on [Page 68](#).

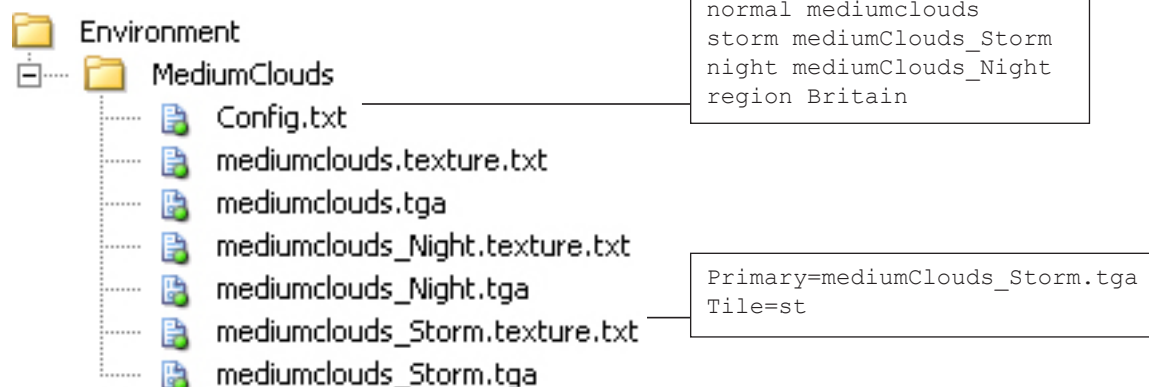
SKY

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Environment\

Sky is generated from three source images.

Typical directory structure for a custom sky should be:



SCENERY

Default location:

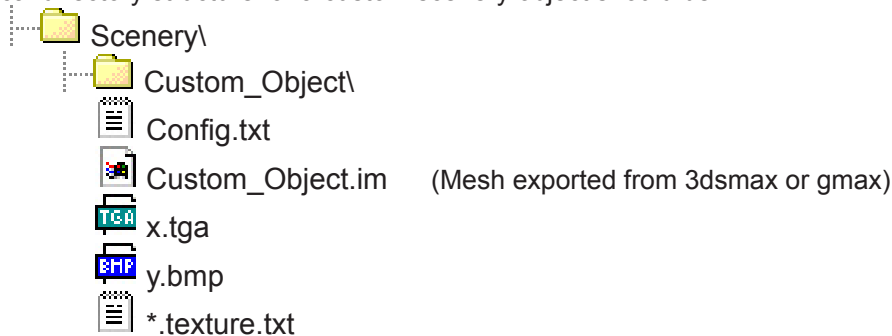
C:\Program Files\Auran\TRS2004\World\Custom\scenery\

Scenery objects can vary greatly in size and appearance. It is recommended to keep the models as simple as is reasonable regarding texture and polygon usage.

Note: Animation Events

Sounds events and generic events can be linked to an animation key-frame to give great control over sound and script timing for industry and scenery assets. Refer to [Page 104](#) for information.

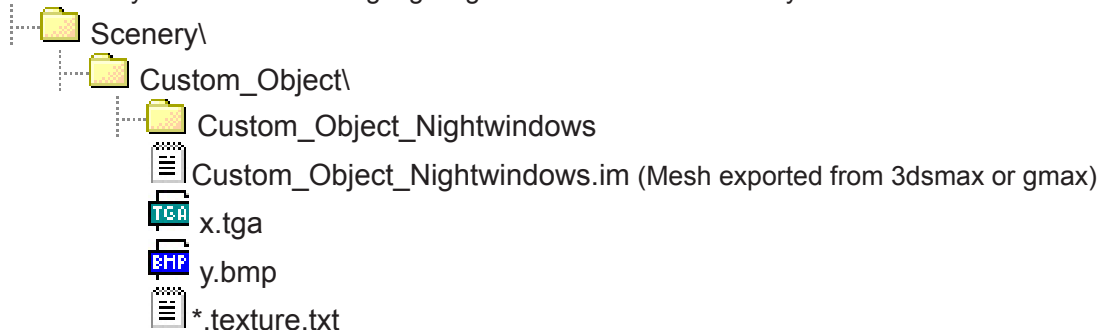
Typical directory structure for a custom scenery object should be:



SCENERY OBJECTS WITH LIGHTS AT NIGHT

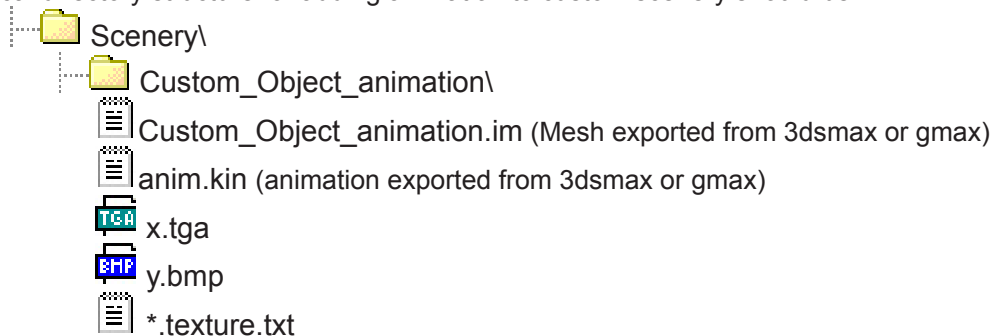
Objects just as buildings or signs can be made to appear to be have lights on at night. A model that contains only the lit areas of the object can be exported into a subdirectory.

Typical directory structure for adding night light effect to custom scenery should be:



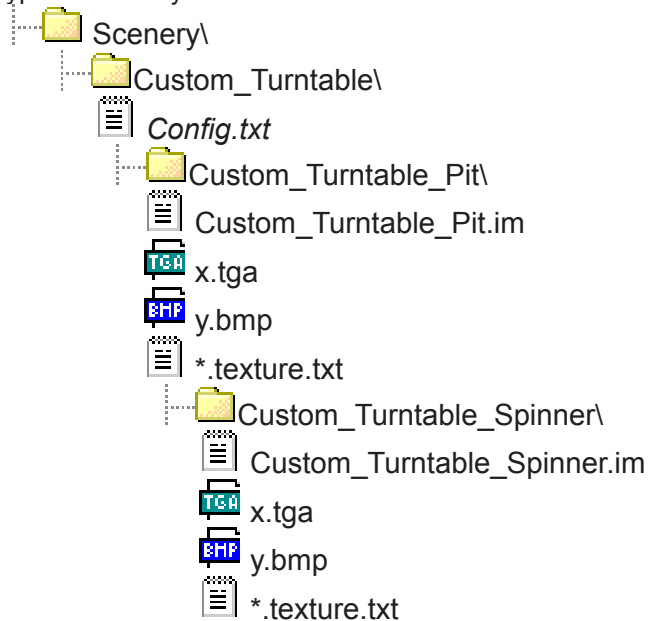
SCENERY OBJECTS WITH ANIMATION

Typical directory structure for adding animation to custom scenery should be:



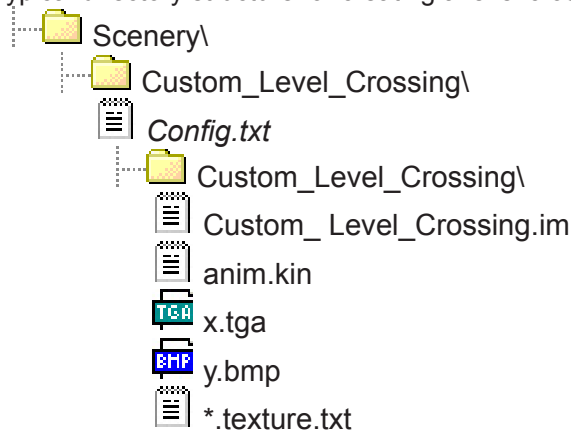
SPECIAL SCENERY OBJECTS - TURNTABLES

Typical directory structure for Turntables should be:



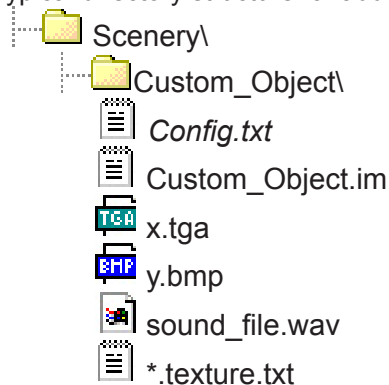
SPECIAL SCENERY OBJECTS – LEVEL CROSSINGS

Typical directory structure for creating a level crossing object should be:



SCENERY OBJECTS WITH SOUNDS

Typical directory structure for adding sounds to custom scenery should be:



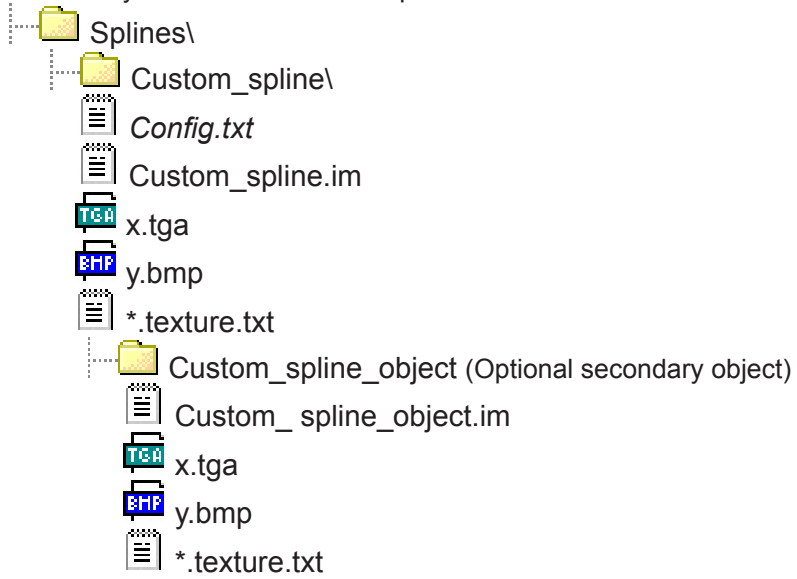
SPLINES

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Splines\

Splines are a useful way of making things like fences and roads in Trainz.

Typical directory structure for custom splines should be:



TRACK

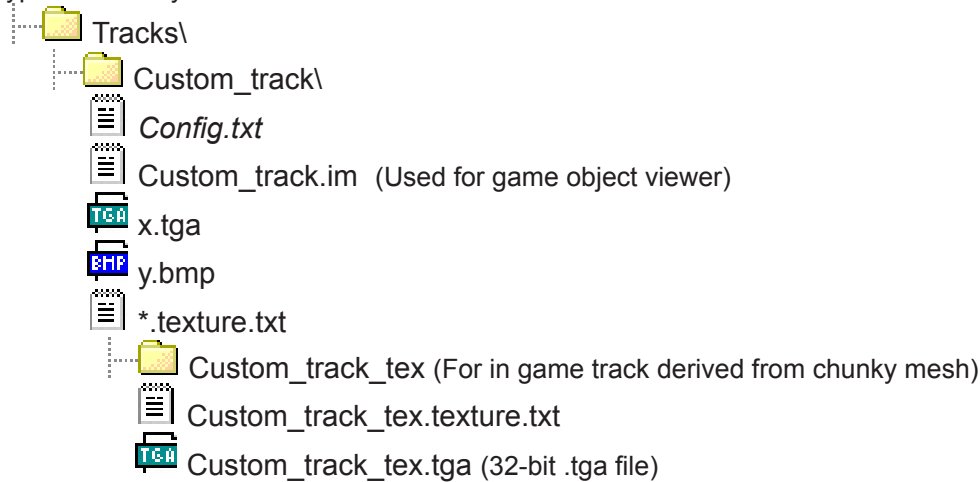
Default location:

C:\Program Files\Auran\TRS2004\World\Custom\Track\

Track folder is used for rails, bridges and tunnels.

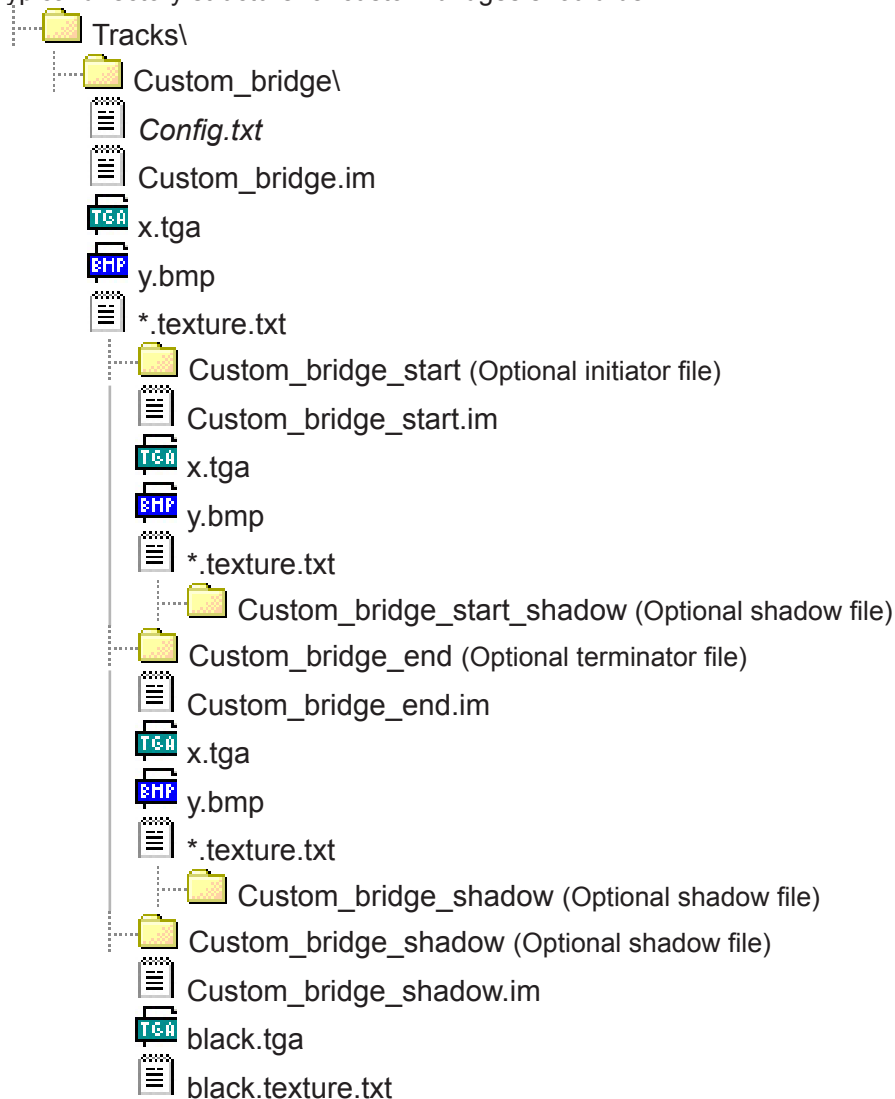
Rails

Typical directory structure for custom track rails should be:



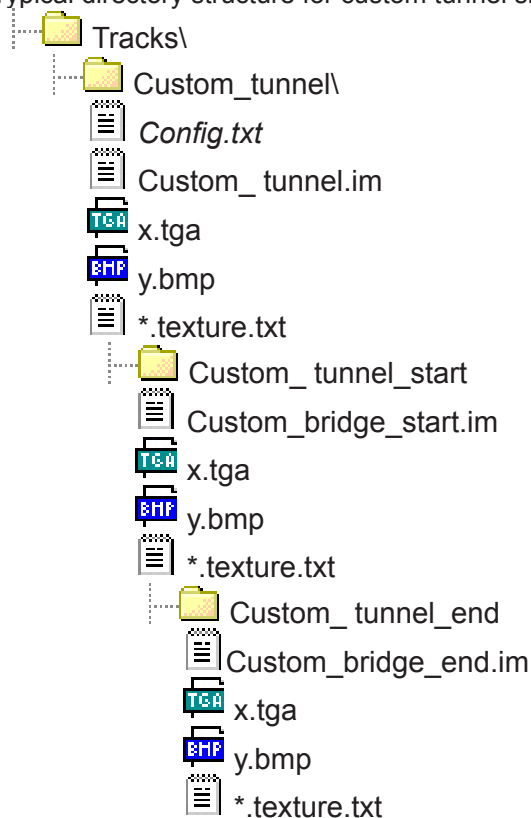
Bridges

Typical directory structure for custom bridges should be:



Tunnels

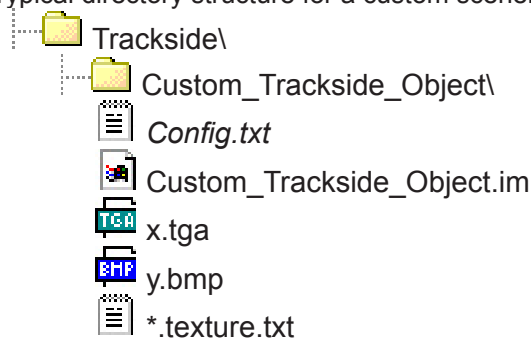
Typical directory structure for custom tunnel should be:



TRACKSIDE

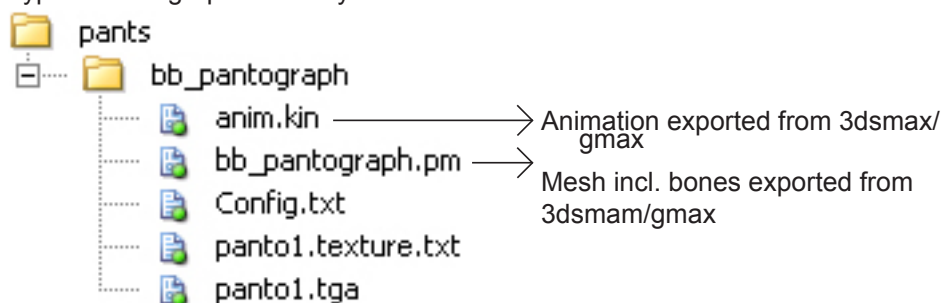
Trackside is used for special scenery objects that can be placed on or near the track, such as signals and speed limit signs.

Typical directory structure for a custom scenery object should be:



PANTOGRAPH

Typical Pantograph Directory Structure



CHAPTER 8

Modeling Guidelines

The purpose of this chapter is to assist the production and installation of custom TRS assets. We assume that third party developers have a sound knowledge of 3dsmax or gmax and therefore only give references to model requirements, rather than a modeling tutorial.

3DSMax/gmax Interface with Trainz

The interface between creating the models and exporting to Trainz can be an area of frustration, when the model does not show in Trainz, or animation operates incorrectly. This section discusses how to set up 3DSMax/gmax, place things correctly and the export requirements. It is not a tutorial on how to make models, manipulate shapes or to map textures.

1. 3DSMax/gmax Initial Setup

You should first set up 3DSMax or Gmax with default settings, both programs have similar settings. The units are easiest to work with when set to metric, some experience problems with animation in imperial units.

Set the menus as follows:

1. From the Customise .. Preference Settings .. General, set the System Units Scale
2. From the Customise .. Units Setup, set the Units
3. From the Grid and Snap Settings ..Home Grid, set the grid to a very fine value, say 0.01 metres. This allows you to have a fine grid when you zoom in, and allows you to snap objects to the grid points (this option may be turned on or off in 3DSMax/gmax). The grid will not clutter the screen as it only shows a grid density suitable for viewing as you zoom in or out.
4. From the Customise .. Preferences .. Files, some boxes may already be ticked, but the following settings are useful.

Increment on save, after first saving the file with a name of your choice, subsequently using the save button will automatically create a new saved file, incrementing the file number. For a file called bridge, this gives brideg001, bridge002 etc as you Save the file, you do not need to use Save As. Save often, when you have finished the model, you can remove older files later on, but if you have a problem with the model you can go back a few saved steps and start again.

It is very important to check the Auto Backup. If 3DSMax or gmax crashes (and it does when you are trying to undo too many steps when mapping, or typing a material name in some of the dialogue boxes in gmax) you can find three files called Autobak001, 002, 003 in the Autobak directory.

3dsmax cycles through these numbers and overwrites them. Find the directory, look at the time stamp for the most recent file, open it in 3DSMax or gmax and you will only have lost up to 5 minutes of work (whatever time you have set). You may have to "Show All Files" to see the file, when trying to open it from the Autobak directory.

5. From the Customise .. Configure Paths .. General you can choose the paths for your files. It is useful to set the Export path to the full directory of the Custom Trainz

directory. It is simple to then navigate to the particular model directory from there, without navigating from the C: directory every time you export a file.

An easy way to have consistent 3DSMax/gmax settings is to load a previous model and then select the "new file" option to begin a new model, retaining the settings from the previous loaded file.

One issue with 3DSMax/gmax is "saved file corruption". Your saved files naturally get larger as you work, but a reasonable files size may be 100k to 400k, depending on the model. Sometimes the next file save becomes corrupted and the file size will jump to 2 or 3mb.

While it does not seem to have any effect on the file contents, you will notice this effect by the greatly increased save and load time of the file. This has been recognised by Discreet, but presently they have no solution to the problem. If possible you should go back to a previous uncorrupted saved file and try again.

Do not merge any object from the corrupted file into a new file, it can corrupt the new file as well, and inflate the size. If you were also using the Increment on Save option, you can then end up with many large files, using up valuable disk space.

2. Merging and Exporting

If you have a completed model, or part, that you wish to use in the currently open model, use the File... Merge function to select and load the part/s. You will have a list to choose from for the parts you wish to merge. The parts will be loaded in the same relative location to the origin, as when they were originally saved. Be aware if they were hidden in 3DSMax/gmax in the original, they will be hidden when merged.

Exporting the model to Trainz requires the use of the Asset Creation Studio for gmax or the export plugin for 3DSMax. Use the Asset Creation Studio to load gmax, do not use the gmax icon or menu to load it, or the Trainz export option will not be enabled.

Make sure all surfaces have been mapped with your materials or error messages may indicate that certain parts of the model cannot be exported.

You should save the model before any export, a good practice to make sure you have a saved file in case of computer problems. You must save the file before the first export, or 3DSMax/gmax will not know the correct path for file location and materials when you try to export, and an error message may result.

When exporting, have no part of the model selected, so you can be sure of what is exported. If you want to export only a part of the model, select those parts and use the Export Selected option. Alternatively, hide all parts that are not to be included.

See the section on Animation Requirements for additional issues with exporting animated models.

Under Tools, use the Resource Collector option to automatically find all the textures for the model and export them to the model directory.

The exporter creates an additional file, for example model.gmw. Files with the gmw extension are not used by Trainz and should be deleted.

When you have finished developing a model, before you package it using the Content Dispatcher, a good practice is to clean out the Custom directories and export the model and all the texture files a final time. This clears out any unused texture files and texture.txt files that were generated during development and are not used in the final model. Leaving unreferenced files in a model can give errors. In clearing the directory make sure you do not delete the config.txt file or any readme file you have created.

If you develop a large number of models you may find that keeping the 3DSMax/gmax file in a suitably named directory along with the texture files for that model, makes it easy to find the exact files used for the model later. Commonly a sub directory under 3DSMax/gmax, such as "Scenes" might be used for the purpose. Use simple names for directories and files, to minimise typing errors in the config.txt file.

3ds is a widely used graphical file standard, and 3DSMax can export this type of file, which can then be imported and used in gmax. You may find that some things like attachment points may not be exported. It also truncates the material file names to 8 characters. When imported into gmax you need to rename the materials to include the original longer file name characters for the materials to be loaded correctly.

Gmax does not export 3ds format, so it is difficult to transfer a gmax file into 3DSMax.

3. Animation Requirements

Adding animation to models requires very specific steps and standards in 3DSMax/gmax. The following are a few key points in having animation export and work correctly:

1. Animation is set up using helper (dummy) points using the b.r. naming notation. It is usual to have a main point (b.r.main or b.r.base or similar) as the main reference point, often placed at the origin. All animation and objects must be linked to this dummy, directly or through other dummies connected to b.r.main.

2. Place the b.r. dummies in top view, and move them to their correct locations, for the individual movement of parts of the model. They are usually placed at the rotation point of the part. Placing a dummy in top view sets the orientation of the dummy axis correctly at the start. Normally, do not rotate the dummy, this misaligns the axis. If the dummy is rotated, the axis must be re-aligned to the World axis.

3. All parts must be placed correctly at the start of the animation, key frame 1. **The axis of all parts must then be aligned to the World axis before commencing the linking of parts.**

5. Dummies are linked to the b.r.main, and the objects are linked to their respective dummies. When this is done, do not move or adjust the parts, unless you are recording the movements as part of the animation (the red animation box is turned on).

6. Do not use groups in animation, they will not export. You can however choose a number of separate parts and link them to a common dummy in one step.

7. Every part of the model must be linked to a dummy, static parts must be linked to the b.r.main.

8. Apply animation to the dummies only, not the objects. Do not move the dummies unless you are recording the animation (see 5 above).

9. All relevant dummies must be exported with the animated objects. If you have two separate animations for the same model (animations in separate subdirectories for example) the common b.r.main must be exported for each animated model part. Failure to include this may give a message "nothing to export". **Again, once you enter any dummy in the model and export it, even before you have started any animation, all parts must be linked to a dummy, and the dummies linked together, or parts will not show in Trainz.**

10. An event file made be used to start and stop animation with the help of triggers and script files. The event file is a simple text file saved with an .evt extension. When exporting the animated model, you will be queried for an event file. You may point the exporter to the appropriate file, and the commands from the file will be incorporated in the animation export. See the event file section in the Content Creation Guide for details.

11. Some difficulty may be experienced with animation in imperial units, or mixed units if care is not taken. Metric units are recommended.

12. The Trainz exporter will only export translation and rotation in animations, not scalar. An object can be moved and rotated but you cannot change the size of the object.

13. Bones may be used for smooth animation in Trainz. They represent parts of the model, aligned to the world axis, and linked. A bone linkage constrains the motion, for example, moving a hand moves the wrist, elbow and shoulder to suit.

Bones must be linked to the b.r. dummies, the objects themselves are also linked to the dummies, and only the dummies are animated, not the bones or objects. The animation of the dummies is constrained by the structure of bones.

4. Attachments

Attachments are the means of specifying how sub meshes or other effects are placed or attached in a model. An attachment point is located in the model and reference is made from the config.txt file, to define its use and function.

Attachment points use the a.name convention, to be recognised in Trainz. For an attachment point to move with an animated dummy, there is a special naming convention for the point in 3DSMax/gmax, e.g. *a.r.dummy/a.name*. Refer to the Content Creation Guide for further explanation of this option.

Attachment points should be created in top view, and the orientation of the axis will be consistent. Sometimes the attachment point may not be facing the correct direction, and has to be rotated (realigned). The Content Creation Guide will give advice on any special orientation required for different attachment points, for track, names, bogeys, coupler points and effects.

Be aware that the point itself must be rotated correctly, not the axis of the point. The Hierarchy...Affect Pivot Only option may be turned on to determine the orientation of the point, it must then be turned off before rotating the point (click on the Affect Pivot Only box to toggle on/off). Turn on the axis information again to verify the rotation of the point has been done correctly.

a.bog, a.limfront and a.limback are some of the important attachments for rollingstock models. As these have specific placements, it may be helpful to save a file of these points as a template, so you can merge it with your new model. With a few adjustments, you can quickly incorporate the template into the model and be sure the point names and orientation are correct.

5. General Modeling Notes

A few general notes on other aspects of 3DSMax/gmax when creating models may be helpful:

Hidden Surfaces and Polycount:

Any surface that will not be seen in Trainz should be deleted to reduce the polycount and the effect on performance. When you initially create a number of shapes, the default settings usually has more segments than you need.

For example, a cylinder by default has 5 height segments and 18 sides. These should be reduced to suit the purpose, many cylinders need only one height segment unless they are to be bent, and small handrails need no more than 4 or 5 segments, and sometimes 3 can be used (or hidden surface also deleted), especially if a smooth modifier is applied to blend the texture around the corners.

Planes default to 4 by 4 segments, this is not usually required. If the 3DSMax/gmax window is set to Smooth + Highlights, the segments may not be noticed. Change to Wireframe mode to see how many segments are used in

an object, and reduce segments to the minimum suitable for the purpose.

Shortcut keys:

There are a number of shortcut keys that are useful, refer to the Help display for 3DSMax/gmax for a guide. A particularly useful one is the function key F2. When in Edit Mesh mode and using the Polygon option to choose surfaces of an object by clicking on them, the F2 key will toggle the color of those surfaces to show you exactly which ones you have chosen.

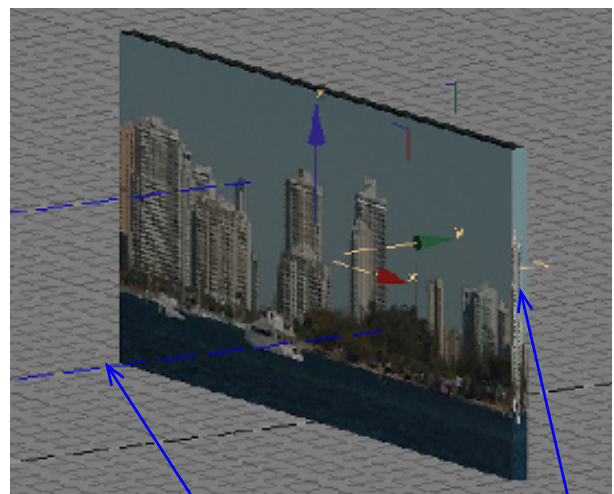
The Ctrl key is used to choose a number of surfaces or polygons together, for a complicated object.

Two sided textures:

For surfaces like planes that can be seen from two sides, ticking the 2 sided box will make the one texture show on both sides. Be aware that this can increase the effective polygons that have to be displayed in Trainz, however Trainz will only have to "paint" those surface that are towards the viewer, so the problem is somewhat reduced.

If you wish to have two different textures on the opposite sides of a plane, you will need to make two separate planes separated by a small distance and each textured with the appropriate texture. Make sure the Normals face the correct direction (see below), or the plane will be transparent. In this case, do not tick the 2 sided texture box for either texture.

The example below shows a billboard which is to have a different picture on each side. A thin box is the basis for the model.



normals show in blue
(see below)

remove the box sides,
or texture the sides separately

Normals:

Normals are the direction of the primary surface of an object and effect the color displayed in Trainz. Normals may be turned on in 3DSMax/gmax and show a blue line from any selected surfaces (polygons). The normals should be aligned for surfaces facing the same direction. If planes are cloned (then rotated) or mirrored, the normals can be facing in the opposite directions. These planes show a mixture of light and dark texturing in

Trainz and the misaligned normals should be “flipped” in 3DSMax/gmax to align correctly with other planes.

Boolean Operators:

The boolean function is often used to cut holes in a model (a window for instance). For a sequence of cutting operations, the model should be converted between operations, to an editable mesh, it can cause problems if this is not done. The cutting action can also cause very long thin polygons in the surface, these can be difficult to texture map. Removing the polygons and replacing them with ones of a more regular shape can make the mapping easier and reduce the actual number of polygons to make the surface.

Reset XForms:

Cloning and mirroring can result in an object appearing “inside out” or appearing hollow in Trainz. This is a result of the pivot point alignment being changed while processing. It cannot be fixed by flipping normals.

Use the Reset Transform utility to align object pivot points and bounding boxes with the World coordinate system. To reset an object’s transform, select the object, and on the Utilities panel, click Reset XForm. In the Reset Transform rollout, click Reset Selected.

You can collapse the object modifiers to absorb the corrected rotation and scale values into the object mesh.

Hiding and Grouping:

The hide function allows you to display only the parts of a model you want to work on. It is useful for complicated objects made up of individual objects, so you can see what you are working on and when exporting certain parts only of a model.

Grouping is also useful, allowing easier selection of a number of objects to transform or hide. If a group is opened, the hide function is disabled until the group is closed. Grouped objects cannot be linked to dummies in animation, the grouped objects will not show in Trainz.

Perspective and User Views:

These allow you to rotate around an object and see the object from different angles. Perspective view centers on the centre of the view box for zooming, so you have to move the object often to zoom into different parts. It also has a clipping box, which makes the object vanish as you zoom closer.

User view allows you to zoom in and out to any point on the object easily, does not clip the view, but it also does not rotate the object about the centre point of view. Each viewing option has advantages and disadvantages for modelling. The mouse wheel is very useful in zooming.

Model Centre of Rotation:

A model should be centred on the origin, as this will be the centre of rotation in Trainz. If an object is set off centre, (the origin is away from the object in 3DSMax/gmax, it will appear very small in the Trainz Surveyor menu selection window.

6. Config.txt File

The requirements of the config.txt file have been covered in the Content Creation Guide for a number of models. The commands recognised by Trainz are known as Tags. A few important points should be mentioned here:

1. The config.txt file must not include any formatted code or symbols. A simple text editor such as Notepad is to be used, the file must be saved as UTF-8 code, not ANSI. This encoding option is available from the save dialogue box. Do not use a program such as MS Word that can introduce unwanted formatting including non standard quotation marks.
2. The order of the lines for most general tags may be varied. Of course, certain tags must be grouped within their appropriate sections (mesh tables, effects, processes and queues). It is useful to have the KUID of the model as the first line, and group the mandatory entries together (categories, Trainzbuild, region).
3. Brackets and quotation marks must be matching pairs (the same number of left and right facing brackets for instance), or tags and information will not be interpreted correctly.
4. You do not need to include blank entries for the kuid-table or the obsolete listing. Use these only where necessary, Trainz does not need to process additional blank entries.
5. At the start of a line, any text or symbol that is not a recognised tag in Trainz will be ignored. If you mistype a tag name, Trainz will jump over this line. Make sure names are entered correctly, with no unnecessary spaces, correct hyphens or underscores and full path names as necessary.

The jumping of unrecognised lines is useful for adding comments. On a new line type rem, (for remark) followed by a space and comment text, for example:

rem This section defines the passenger processes.

6. The description entry uses a single pair of quotation marks, do not include additional marks within the description or the entry will be truncated. The Description is displayed on the Download Station so make it informative, perhaps including what the model is called and under what directory it is to be found in Trainz. This will assist a user in finding the model in the Surveyor menu.

7. The Content Dispatcher modifies the config.txt file when the model is packaged:

- It places an apparent blank line at the top. This line contains hidden code used by the Download Station process, and should not be removed or the config.txt file will not function;
- It tabs the entries across the page and inserts quotes around descriptive words. While the tabbing makes the

file hard to read sometimes, it can improve the readability for the bracket symbols, making it easier to match pairs;

- it adds the licence entry from the Content Dispatcher, so do not manually enter any licence details in the config.txt file or you will have duplicate entries;
- it performs some basic error checking and may indicate if you have not included necessary mandatory tags.

7. Problems with Model Exports

Some suggestions for common problems in having a model export to Trainz:

1. The object shows in Trainz but has white surfaces, no texture:

- the texture file is not a recommended size;
- it has been saved as a compressed file;
- the reference name is spelt incorrectly; or
- the texture files have not been exported to the Trainz Custom directory (use the Resource Collector).



2. Some faces of the model are invisible in Trainz:

The faces or surfaces have a single sided texture and the normals are facing away from the viewer.

3. Some surfaces of planar objects show darker colours (in shadow) when lit by the sun in Trainz:

The sun side of the object shows dark, and the unlit opposite side of the object shows a lighter colour if the normals are facing away from the viewer for a 2 sided textured object.

This can happen when a plane object is copied to the opposite side of a model, plates on a steel bridge for instance, and the normals have not been flipped to face outwards towards the viewer, on that side of the object.

After selecting the face, and clicking the Show Normals box, use the Normals: Flip option to change the normal direction to align with other normals of the model.

4. Building walls which include transparent windows are see-through:

A transparent texture with an alpha channel or opacity map has been applied to the windows as part of the wall

texture. Window transparency must be applied to window planes separate from the main building wall and a separate texture must be used for the window and for the wall. Do not add an opacity layer to the wall texture.

5. The animation does not work in Trainz:

- the anim.kin file has not been exported;
- parts of the model were hidden when exported;
- all the dummies were not included in the export;
- the config.txt file is incorrect, particularly with reference to names of files, missing lines in the file, incorrect matching brackets or quotes;
- the animation-loop-speed 1 tag has not been entered in the config.txt file; or
- the default modelling units are not consistent.

6. The animation is working but the animated parts are scattered over the landscape in Trainz:

- the axis of the parts were not aligned to the World coordinates before linking and animating; or
- the objects have been moved after linking, but the movement was not recorded as part of the animation, and the axis reference has changed; or
- parts are linked to the incorrect dummy.

Unlinking the parts and dummies, and re-aligning the axis does not always fix the problem. Often the dummies have to be deleted and replaced, with all the aligning, linking and animation redone.

7. The lettering on a sign using the a.name attachment point option is not visible or is facing the wrong way:

The axis of the attachment point is not facing the correct direction. You must rotate the attachment point, not the axis, in 3dsmax/gmax. Refer to the Content Creation Guide for the correct method and orientation.

Orientation can also be a problem with corona visibility.

8. Deleting a model in Trainz sometimes leaves the track attachment points behind, or attachment point changes do not show.

When developing a mocrossing type object, you may have changed the location of attachment points in 3DSMax/gmax. If the model has already been placed in Trainz, these changes do not show unless the original model is deleted and replaced.

When a model is deleted, sometimes the attachment point circles remain. Change to the Track menu in Surveyor to delete the obsolete attachment points.

POLYCOUNT

3D STUDIO MAX AND GMAX

MODEL GUIDELINES:

This page contains an outline of the mesh asset polycount guide for the various types of assets. Of course being a 'real-time' 3D engine we strongly suggest you keep the polygon count to a minimum.

In other words - in what environment is the asset being used? Do you have a single house in the middle of a desert or is the house surrounded by trees, power lines, other houses and buildings, a train track and a 100,000 polygon full train consist zipping by every 10 minutes?

Based on this thought, then consider the following:

User system variations

User system Performance

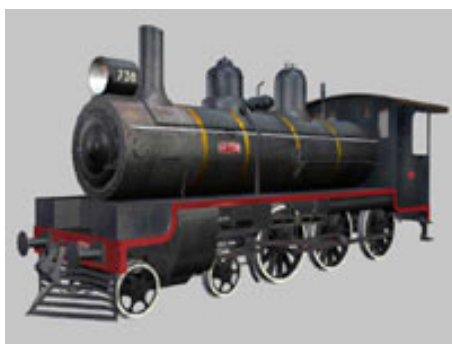
If you need a ladder, use alpha maps, if you need a steel structure, use alpha maps, if you want to model a signal pole only use a 5 sided cylinder, if a locomotive has lots of pipes and handrails, make the pipes 3 or 4 sided (and use the same smoothing group to get the pipe effect).

The following table will give you a guide for creating your models. Slight variations to these are possible but it is not advisable for example to use 17,000 polygons for a locomotive body.

MESH POLYCOUNT GUIDE

KIND	MAX POLYCOUNT	NOTES
Locomotive Interiors	8,000 each	Diesel or steam loco's. Including all levers, dials and animations, & bonnets (hoods for our American friends).
Locomotive Body (diesel)	9,000 each	This should be sufficient for all diesel loco's - excluding bogeys, pantographs etc. For example, the 'QR 2100' body currently in Trainz has less than 5000 polys. Less is always better!
Locomotive Body (steam)	12,000 each	This should be sufficient for very detailed steam locos - excluding bogeys, etc. (see images at bottom of page).
Locomotive Bogey (diesel)	2000 per truck	
Locomotive Bogey (steam)	5,000 per driving wheel set	Including all rods and animated parts. Less is always better! Should be sufficient enough for most steam bogeys.
Pantographs	1,000 each	On average pantographs will have about 400-700 polys.
Passenger Rolling stock	2200 each	This should be sufficient for a passenger car with window cut-outs, a low poly interior with seats (visible from the exterior), and animated doors.
Hopper Rolling stock	1400 each	This should be sufficient for most Hopper cars, inclusive of load and animated doors.
Flat, box and tank Rolling stock	1100 each	This should be sufficient for most cars, inclusive of load and animated doors where applicable.
Rolling stock Bogeys	400 each	Rolling stock bogeys must be kept to a minimum. Make the bogey sides basic, even alpha mapped only.
Level Crossing	350 each	Based on a simple road/track crossing with animated boom-gates and signals.
Single Caternary	200 each	Based on per pole structure.
Semaphore Signal post	300 each	Ladders should be 2-sided alpha mapped.
Coloured light Signal post	300 each	Ladders should be 2-sided alpha mapped.
3D Passengers	140 each	
Typical House	150 each	The 'Australian houses' currently in Trainz average about 150 polygons.
Typical Large Building	300 each	

KIND	MAX POLYCOUNT	NOTES
Other Scenery Objects	Varies	Just don't go overboard! Put the detail in the textures. Remember not to go overboard with the texture sizes either! 512x512 pixels is too big for a house. The UTC German Lumber Yard for example, including it's forklift and scary looking driver, came to only 1714 polys. The logs are 5 sided cylinders, the dominant chimney stack is an 8 sided cylinder.
SHADOW: Loco	700-900 each	Just enough to model loco form (usually including buffers)
SHADOW: Passenger cars	less than 300	
SHADOW: Tank cars etc	less than 500	
SHADOW: Bogeys	less than 100	Simple box and wheel faces are usually enough



The images above are an example of the kind of detail you can get within the polygon boundaries above.

Body: 10,578 polygons (hi-res L.O.D. version)

Front bogey: 696 polygons

Rear bogey: 4018 polygons

TOTAL: 15,292 polygons (body & bogeys)

With respect to L.O.D., this locomotive's polygons can be reduced very easily:

The pipework, valve and gauge details within the cab equate to 4775 polygons.

The pipework and handrails on each side of the boiler equate to 948 polygons.

The coupling and pipe at the front equate to 459 polygons.

This could mean a body reduction of 5882 polygons without any loss of form.

There are more examples of cab interior polygon counts on [Page 51](#).

TRAINS

3D STUDIO MAX AND GMAX MODEL GUIDELINES:

Download Source files from the Trainz Website

POLYGON LIMITS:

Train *body* polygon recommendations (excluding bogies)
Diesel loco = 3500-9000 polygons.
Steam loco = Up to 12000 polygons.

As a general rule of thumb, less is always better! ☺

The front end of the train body should be on the LHS when displayed in the RIGHT viewport.

Train body *shadow* polygon recommendations = Less than 1000 polygons modeled to the same basic shape and 3D space as the body. No attachments are required within the shadow file.

ATTACHMENT POINTS:

(3dsmax & gmax: 'Create' tab, 'Helpers', 'Point')

To maintain correct alignment, attachment points should be created in the TOP viewport.

These are 'points' in 3D space giving information on various aspects of the train as follows:

a.limfront

- Marks the front of the train, used for coupling
- Should be roughly the same distance from origin as a.limback
- Bogies can be further forward than a.limfront if desired
- Determines the forward headlight position
- Height above origin (or Z) = 0.89m (2' 10.8")

a.limback

- Marks the rear of the train, used for coupling
- See a.limfront
- Height above origin (or Z) = 0.89m (2' 10.8")

a.bog0

- Front bogey attachment
- Used for positioning the train on the track
- Positioned at absolute centre of front bogey

a.bog1

- Rear bogey attachment
- Used for positioning the train on the track
- Positioned at absolute centre of rear bogey

a.bog* (2, 3, etc)

- Any other bogey attachments

a.exhaust* (0, 1, etc..)

- Smoke generator attachments (where needed)

a.light* (0, 1, etc..)

- Light "corona" attachments

a.cabfront

- Attachment point for the front cabin of a loco
- Located at the centre of cabin

a.cabback

- Attachment point for the rear cabin of a loco. Use this for dual cab locomotives.
- Located at the centre of cabin
- Front/back cab toggled using the 'Alt C' key when using the internal camera mode.

a.pant* (0, 1, etc..)

- Attachment point for pantographs (where needed, i.e. Electric locos)

a.driver* (0, 1, etc..)

- Attachment point for driver mesh (0 is used for the first driver, 1 for the second driver in dual cab for instance). Currently, only a.driver 0 is supported.

a.outsideview* (0, 1, etc..)

- These are located external of the loco body mesh.
- The camera is positioned to face the *negative Y* direction of the attachment.
- Toggled using [and] using the internal camera mode after default interior camera view(s).

In addition to these, you may add any other attachments so long as they use the *a.name* naming convention. These can be used as steam or smoke points, or as an attachment position for another mesh or animated mesh.

All additional smoke and mesh attachment points are referenced through the Loco's config file, smoke through the smoke fields, the mesh attachments through the mesh table field.

TRS has the ability to allow it's rolling stock to pick-up and deliver commodities (or products) to the various industry assets. In the coal hopper for example, the load mesh is a simple animated mesh, that is tied in through the config to the product queue values.

Simple carriage cars require only a.limfront, a.limback, a.bog0, and a.bog1.

Another idea for animated attachments is a diesel locomotive roof fan. This would be set-up using *the same* animated mesh inserted at each point. This can be done easily through the mesh-table.

Refer to TRAINCAR EXAMPLES [Page 38](#) for links to downloadable in-game files, documentation and source files of the various types of TRS compatible traincar assets.

Refer to Passenger Asset Tutorial for information on passenger vehicles compatible with the new passenger supported features:

http://files.auran.com/TRS2004/downloads/contentcreation/SP2-Passenger_Asset_Tutorial.zip

TRAIN TEXTURES

The materials are of Multi/Sub-Object type (one M/SO only per model) and we have used UVW Map and Unwrap UVW for texture allocation. Textures must be of following pixel dimensions: 8, 16, 32, 64, 128, 256, 512 and 1024 pixels. Maximum ratio = 1:8 e.g. 64x512

Diffuse Maps: In many cases a single 512x1024 24-bit .tga file is sufficient to texture a locomotive. We recommend not making them any larger than this.

Occasionally an extra texture (say 128x256) can be added.

Reflection maps are supported (16 bit colour .bmp). We generally set train body reflection amounts (in 3dsmax) to 10 and windows to 25.

Opacity Maps (8 bit greyscale .bmp) are also supported to the EXACT same pixel dimensions as the diffuse map.

Where possible, opacity maps should be included as an alpha channel of the main diffuse texture .tga file. A separate opacity map degrades the performance of Trainz as the extra file has to be processed separately.

Reflection and Opacity maps must not be used together with-in the same texture. Reflection and Opacity maps must not be used on digits.

Window opacity is derived from the material opacity setting. See figure right.

LOCOMOTIVE NUMBERING

TRS supports dynamic locomotive numbering for custom content (using alpha-numbers). Otherwise known as 'running numbers'.

Digits are modeled as 6 individual rectangular polygons offset from the face of the Loco body (about 5mm). Digit polygons must be texture mapped using the correct texture naming and alpha-number naming conventions as follows:

If one font type used:

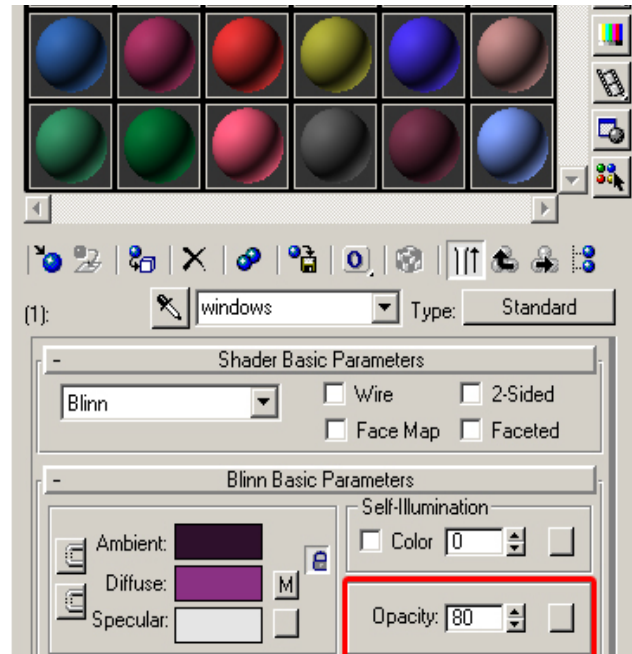
Digit textures (digit_1.tga to digit_6.tga) are replaced automatically with alphanumber textures (alphanumber_0 to alphanumber_9).

If two or more font type used:

Digit textures (digit_1a.tga to digit_6a.tga and digit_1b.tga to digit_6b.tga etc) are replaced automatically with alphanumber textures (alphanumber_0a to alphanumber_9a and alphanumber_0b to alphanumber_9b).

Locomotive numbering in TRS is edited in Surveyor: Trains panel (Train mode) - Edit Properties (the '?' icon).

Refer to Source files for configuration of Loco numbering digit's.



3dsmax/gmax window material

BUMP MAPPED AND SPECULAR MATERIALS

TRS supports bump mapping and specular materials. A nice example of a bump mapped loco with specular values in use is the TRS asset: The SNCF TGV loco.

Bump mapping *

3dsmax 4 + users only (at time of writing).

A gmax exporter update may soon be available pending negotiations with the creator of gmax - Discreet.

For those new to this term, Bump mapping is used to add 3-dimensional detail to an image (using an applied RGB 'Normal' map), without increasing the number of polygons.

Bump mapped materials for TRS requires a new 3dsmax exporter.

Specular Materials *

3dsmax and gmax Users

Adding Specular values to a material is best described as adding 'shininess' to the material. Altering specular values can give realistic material properties to metallic and glassy surfaces.

You can specify specular values from 3dsmax or gmax, but it is quite important for bump mapped materials to have specular values in order to highlight the bump mapping effect.

* Note:

Please download the following zip file for information and set-up of bump mapped and specular materials and the 3dsmax4 exporter: http://www.auran.com/TRS2004/downloads/contentcreation/TRS_Max4_Plugin_Bump.zip

BUMP MAPPING INFORMATION

NOTE: Bump mapped materials for TRS requires a new exporter. At time of writing, only a 3dsmax 4/4.2 exporter update is available. A gmax exporter update may soon be available pending negotiations with the creator of gmax - Discreet.

*The new 3dsmax 4 & 4.2 exporter is available from the same zip that this document was extracted from:
http://www.auran.com/TRS2004/downloads/contentcreation/TRS_Max4_Plugin_Bump.zip*

Bump Mapping Background Theory:

For those new to this term, Bump mapping is used to add 3-dimensional detail to an image (using an applied RGB 'Normal' map), without increasing the number of polygons.

A 'normal' is a vector that points into the direction that a surface is facing (orthogonal to its surface). 'Normal' bump mapping applies 'false' normals to each pixel of a polygon, so that the reflection is not computed in accordance to the 'real' polygon surface, but according to the surface vectors of the normal map.

This results in the bump mapping effect, giving the surface a 3D-appearance that is not 'really' geometrically there.

If the user's graphics card does not support per-pixel bump mapping, then the bump mapping effect won't be seen

Bump Mapping in TRS2004:

For all intents and purposes, bump mapping should only be used on locomotives and only in the hi-res locomotive version if 'level of detail' mesh reduction is being used. (Refer to Level of Detail, [Page 105](#)).

Bump mapping can be used to 3-dimensional detail to rivets, bolts, rust and joints for example. The following example taken from the Auran's Steam locomotive the QR Class PB 15 (figure 1).

QR Class PB 15



Figure 1

DIFFUSE TEXTURE 1024x512 24 bit .tga

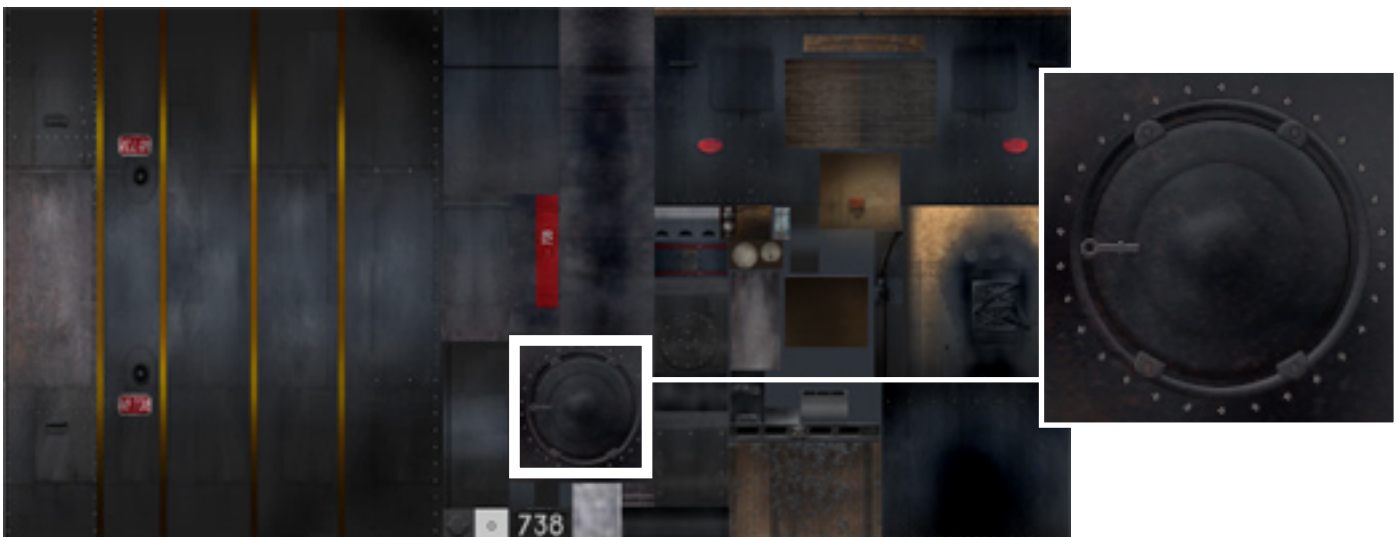


Figure 2

GREYSCALE HEIGHT MAP 1024x512

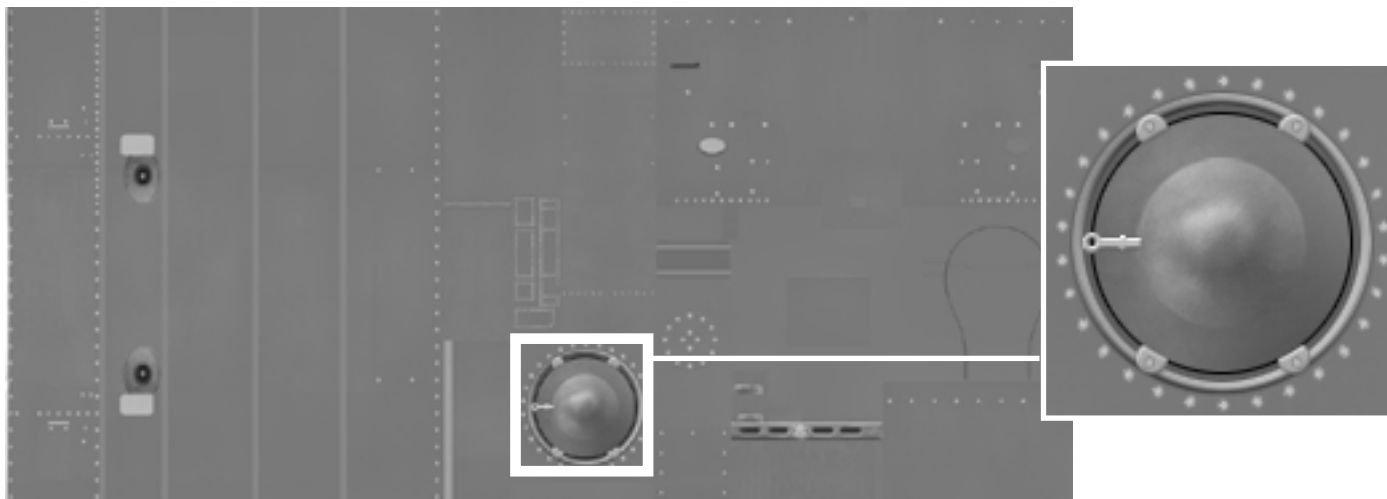


Figure 3

RGB NORMAL MAP 1024x512

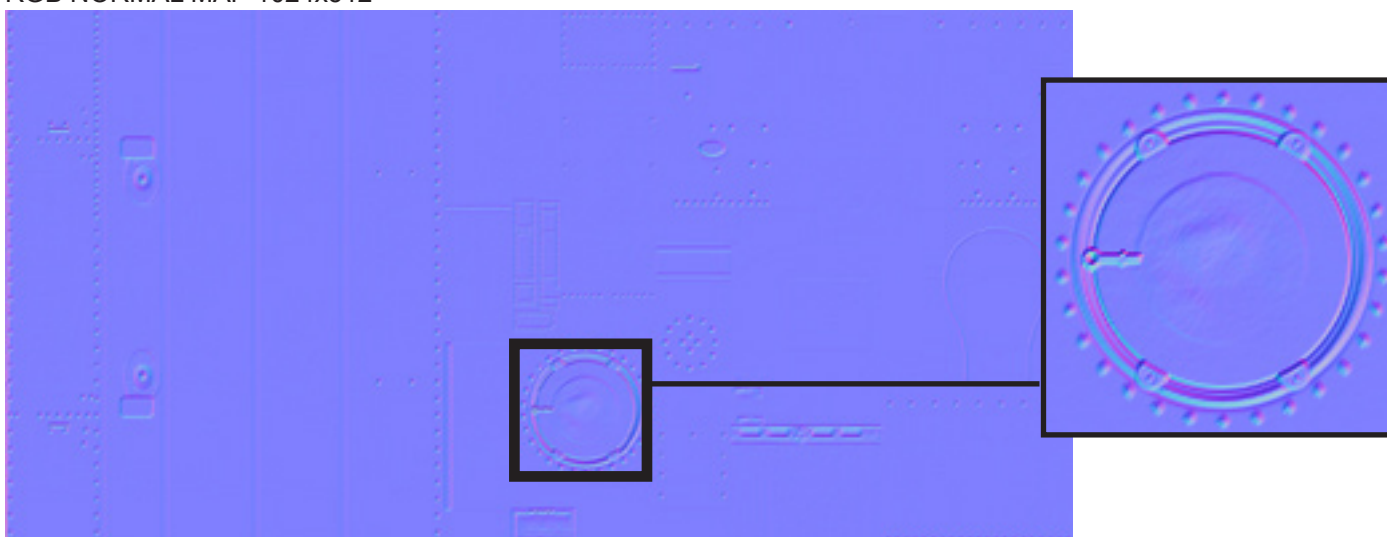


Figure 4

‘Normal’ Map Generation:

This ‘normal’ map was created from the greyscale height map above using a *Photoshop Normal Map Generation Filter* available for download from the [Nvidia web site](#). The plug-in also includes a 3D preview with per-pixel lighting to view the generated normal map. PaintShop Pro 7 users should also be able to use it.

The plug-in generates the normal map by calculating the greyscale contrast. White is high, grey is flat, black is low.

The above normal map was generated using the settings displayed right (figure 5).

Note: Bump map textures will need to be in a pixel ratio of ‘power of 2’ for the filter to function. i.e. 512x512, 256x512, 512x1024 etc. Of course these are the same dimensions as TRS2004 so you shouldn’t have a problem!

Refer to ‘**Important Notes: Normal Maps**’, on [Page 102](#)

PHOTOSHOP NORMAL MAP GENERATION FILTER

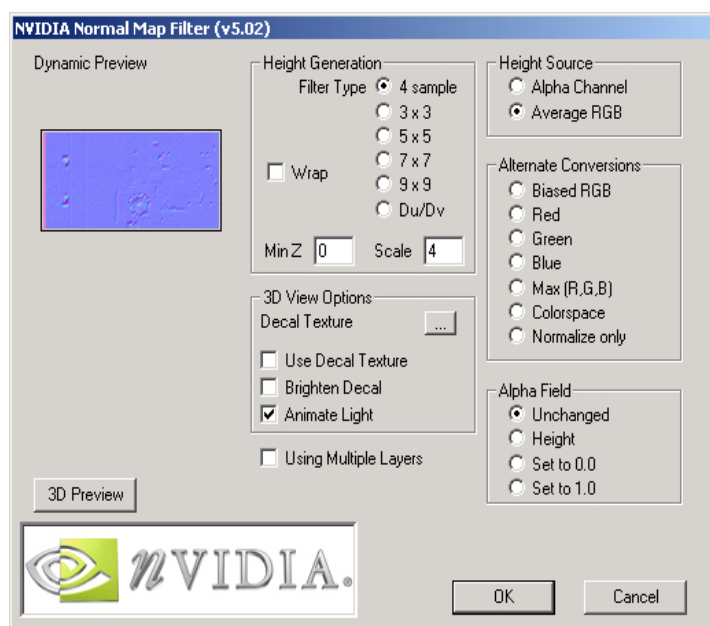


Figure 5

Important Notes: 'Normal Maps'

- Once the bump map has been converted to a RGB Normal Map, do not apply any changes to it like scale, blur sharpen etc. If you wish to alter the normal, apply the changes to the greyscale height map then re-generate.
- The normal map does not have to be the same dimensions as the diffuse map. Normal maps can be smaller to save texture space, larger for finer grain bumps but the later is not advisable at 1024x512. As above, do not apply changes to an already generated normal map.
- At no time should you save RGB normal maps in a 'lossy' file format (i.e. jpg or other compression file format). This is due to the fact that compression de-normalises normals. Use only uncompressed 24 bit tga's.
- Keep all mapping clean. i.e. Don't stretch your mapping co-ordinates. You can get some seriously undesirable effects!

Note:

These settings will also apply when specifying specular levels of non-bump mapped materials. Just remember to make the Diffuse and Ambient colour values pure white (unless you really know what you are doing!).

Emissive values (self-illumination) is also exportable using the above configuration.

Specific material naming conventions need not apply to non-bump mapped materials.

gmax: You can specify Specular, Ambient Diffuse and Emissive settings via Trainz Asset Creation Studio exporter.

Edit the .cfg file as above...

C:\gmax\gamepacks\Trainz\Plugins\JetExporter.cfg
(you just cannot at this stage export bump mapping).

Bump Mapped Materials

3D Studio Max 4/4.2

The setup has altered from previous exporters. This is to allow specular control of bump mapped surfaces (see notes on [Page 99](#) for Specular Control).

After downloading and installing the new exporter, ensure the following configuration:

C:\3dsmax4\plugins\JetExporter\JetExporter.cfg

```
enableWarnings = 1
```

```
[C:\3dsmax4\Plugins\JetExporter\IndexedMeshExport.dll]
```

```
{
  buildNeighborArray = 0
  forceTxtOverwrite = 0
  defaultMaterialColor = 0
  disableautobillboard = 1
}
```

If these values are set to 1, they will override the Ambient, Diffuse, Specular and Emissive settings noted on page 100.

```
[C:\3dsmax4\Plugins\JetExporter\ProgressiveMeshExport.dll]
```

```
{
  buildNeighborArray = 0
  forceTxtOverwrite = 0
  defaultMaterialColor = 0
  disableautobillboard = 1
}
```

```
[C:\3dsmax4\Plugins\JetExporter\AnimationExport.dll]
```

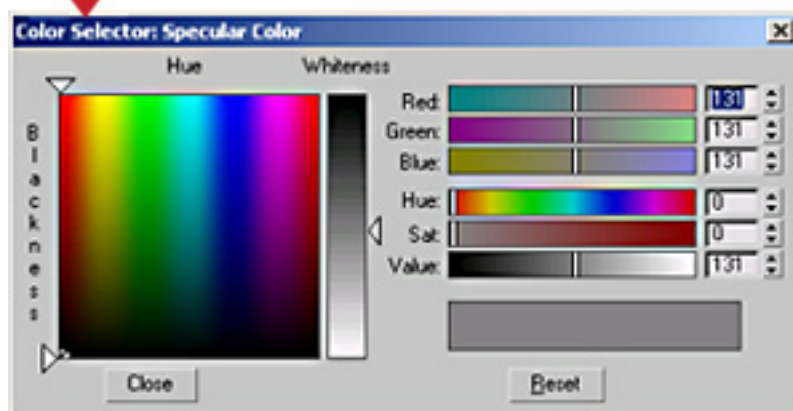
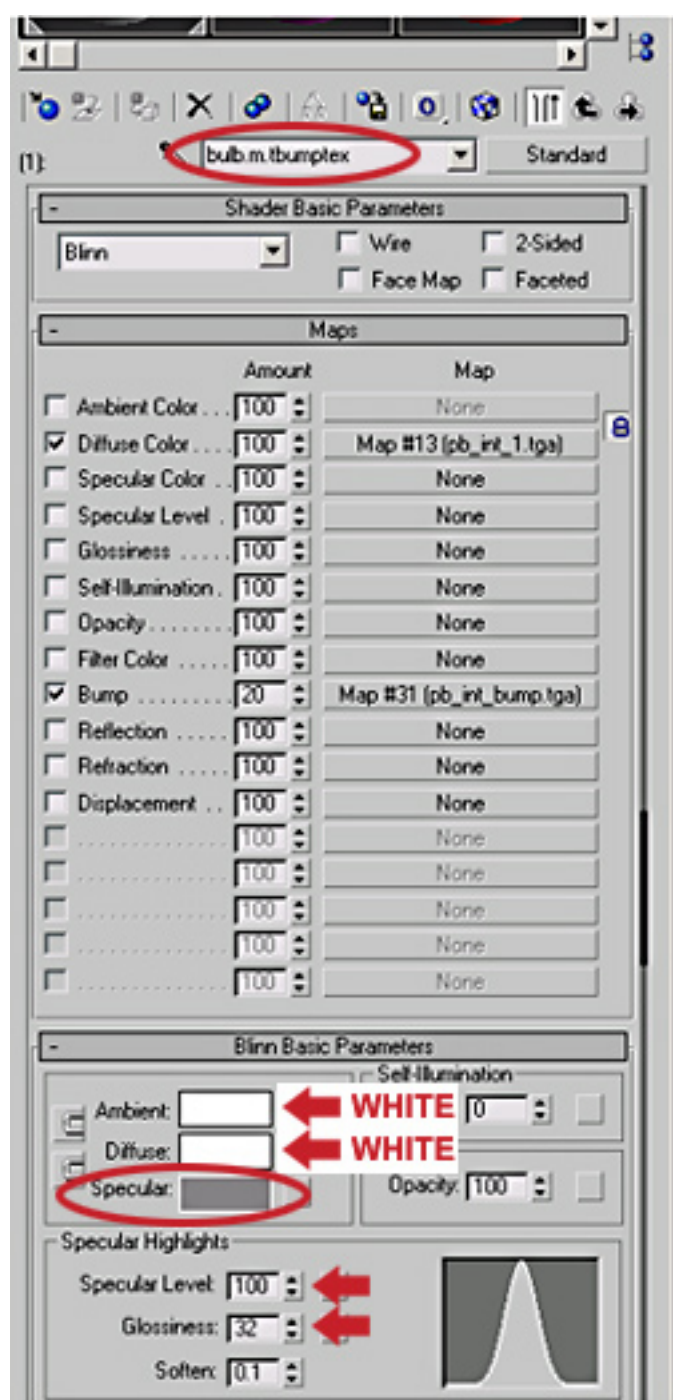
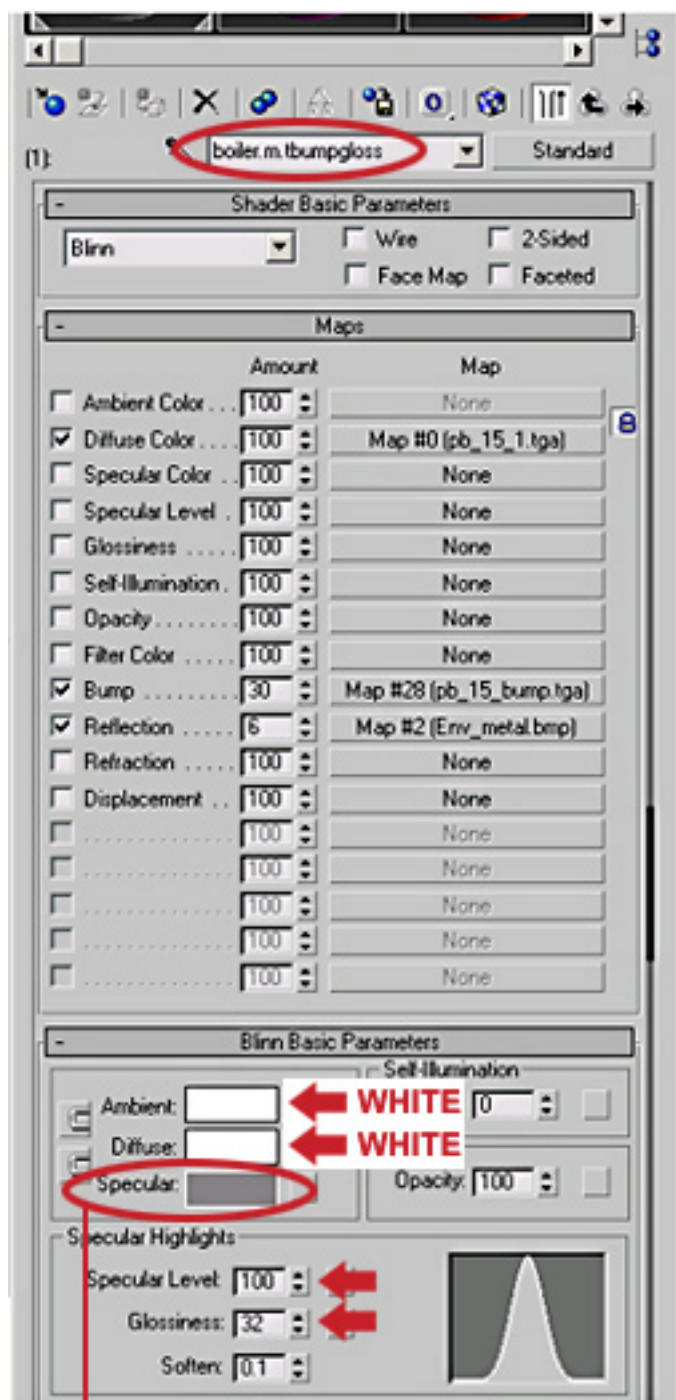
```
{
}
```

Check the drive paths are correct

Material Naming Conventions.

For straight bump mapped materials with or without alpha (*no reflection allowed*) = **name.m.tbump**tex

For bump materials with reflection (or gloss) (*no alpha allowed*) = **name.m.tbumpgloss**



Specular Values:

The specular setting is controlled by the whiteness slider (left). The whiter it is, the higher the value.

- Ensure Specular Level is 100
- For bump mapped materials, the Glossiness value gives a visual representation in MAX only (as TRS forces this value to 32)
- For non-bump mapped materials the glossiness value WILL be exported, you should still use the whiteness slider to determine the specular level. The Glossiness value adjusts now wide or narrow the Specular is.

Textures and Opacity Effects

Trainz makes use of a texture.tga map associated with an opacity texture map to create transparent, translucent and see through effects. This is applied to objects for transparent/translucent windows, ladders, lattice work for cranes, handrails, and catenary, to name a few.

A 24 bit uncompressed .tga texture (diffuse texture) may be created for the object and a 8 bit .bmp map (opacity texture) of the same size is used with the .tga texture to create areas of transparency. The opacity map is predominately black and white, or shades of grey (see below).

Any area on the opacity map that is white, will make corresponding areas on the diffuse map opaque.

Any area on the opacity map that is black, will make corresponding areas on the diffuse map transparent.

Any area on the opacity map that is shades of grey, will make corresponding areas on the diffuse map translucent, depending on the shade of grey.

There are certain requirements in using such maps:

1. Placement in 3DSMax/gmax

In 3DSMax/gmax, when using an opacity map with a diffuse map, the .tga texture is placed in the diffuse colour slot of the Material Navigator, and the opacity map is placed in the opacity slot. Often it is necessary to tick the texture 2 sided box, so the object is visible from all directions, particularly with ladders and windows.

2. Opacity Fade Out

Most opacity maps are primarily black and white. In TRS, requirements have changed for the opacity map. If the map consists only of black and white, a third colour must be added. It is convenient to add at least one pixel of another colour, say rgb 32,32,32 to an area of black.

This is necessary to prevent the object from fading out and flickering a short distance away. It is particularly important with ladders, railings and catenary, models with fine detail.

3. Alpha Channel Use

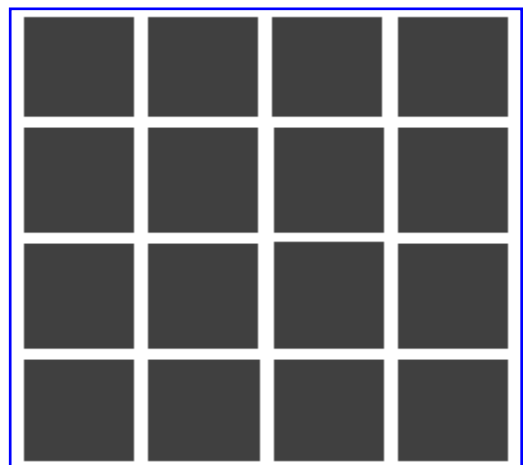
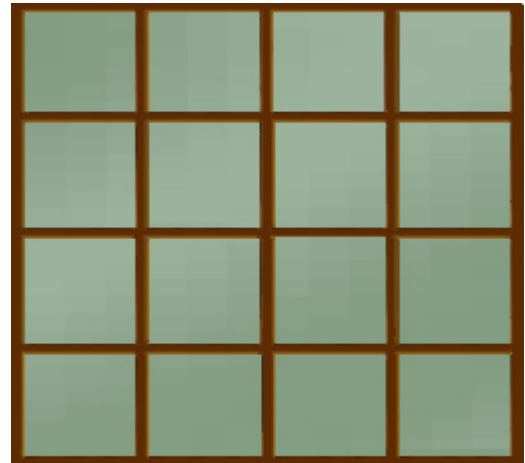
While it is convenient to make a separate opacity map to create the transparent effects, it is better to make the opacity map an Alpha channel of the original .tga diffuse texture. This may be made in Photoshop, Paint Shop Pro or TgaTools2. The primarily black and white Alpha channel is saved (embedded) within the diffuse .tga texture file.

Using the alpha channel procedure is more efficient for Trainz to process, it is quicker to load, than two separate maps. This is the recommended process for loading the opacity effects. Of course there are instances where the separate opacity map is useful (see reflective materials below).

4. Example

The images show a window .tga texture and a corresponding .bmp opacity map.

The window frame will show in the model, because the corresponding area on the opacity map is white. The green glass area will be translucent (show a green tinge) in this example because the “black” areas are actually a grey, rgb, 64,64,64. Once again, it is good practice to combine this opacity file as the Alpha channel of the original .tga texture file.



5. Applying Opacity to Models

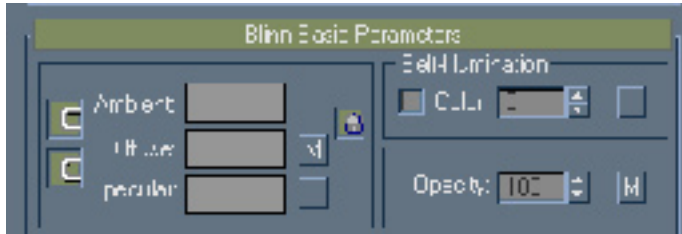
When modelling a building with windows in 3DSMax/gmax, as an example, the windows should be constructed as separate planes and the window texture with the opacity applied to those planes.

This has two effects:

- only requiring a small opacity map to match the window texture, instead of a very large opacity texture of the whole building with a few opacity areas “cut out”;
- more importantly, it is essential to prevent the complete wall being “see through” from different angles in Trainz. An opacity map that is part of a larger wall map can create this “vanishing wall effect”, and flickering.

6. Opacity Settings in 3DSMax/gmax

In the 3DSMax/gmax material editor there is an opacity settings box, where you can change the opacity of a material. This option however will make the whole surface, to which the material is applied, transparent to the degree chosen in the percentage selection, 100% being opaque. It will not allow the fine control of the opacity texturing that the Alpha channel provides.



See the Content Creation Guide for information on texture files.

7. Opacity Interference

If there are two opacity planes (transparency) close together and behind each other, there may be some display issues in Trainz. Current graphic cards are not always capable of determining the depth order of opacity texture in a scene in Trainz. This means that an object behind another object, both using transparency, may actually be shown in front of the foreground object.

An example would be a footbridge using transparency to show the timber lattice construction, being behind a train with transparent windows, the footbridge supports may actually appear in front of the train, instead of behind. This effect can occur on different parts of the same model, but also with separate objects. Be aware of this issue, which is not under the control of the creator.

8. Special Use of Opacity - Reflection Materials

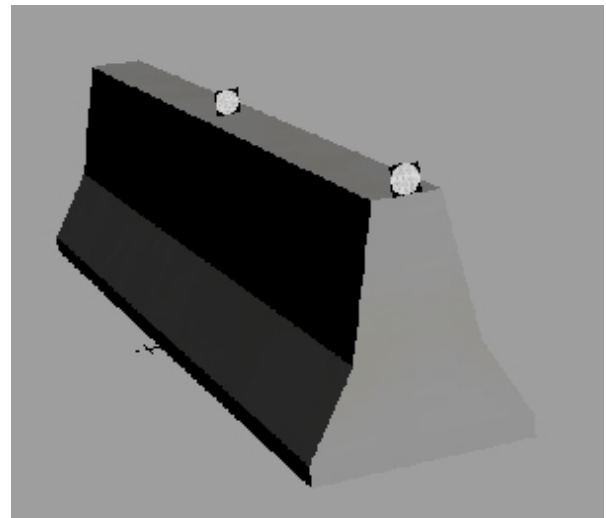
The methods outlined above may be extended to give other lighting effects, using the additional Reflective materials slot in the 3DSMax/gmax Material Navigator.

A nightwindows directory is often useful to provide a night mode for the model, with lights and lit windows. This requires a separate night model mesh, and can be called up as a sub mesh in the config.txt file. Some Kinds such as splines do not support the Nightwindows option. To make a night effect on a spline, certain parts may be specially textured to give a lighted appearance at night.

Three texture files are used, a texture for the day visible object or planes, an opacity file to determine the shape of the lit portion at night, and a reflective texture to give the night colour and attributes. These are applied as one material to the object.

Example:

A road barrier is made as a spline and requires white reflectors at regular distances along the spline, to light up at night. A light.tga file will define the day texture for a



circular reflector, a light.bmp will make the opacity file to be used, and the reflective material can be a single colour .tga or .bmp texture, to be the night colour. All texture files must be the same size, in this case 64 x 64. The image below shows the texture files:



To the left is the reflector.tga, the next is the opacity.bmp, and the third image is the light colour to show at night, this could be a whiter shade, but is left yellow so it shows better in this example image. It is placed in the Reflective material slot in 3DSMax/gmax.

The last image is a circular gradient opacity.bmp texture that could also be used, to give softer edges to the circular night shape.

The yellow colour texture may cover the whole area, but a single pixel of colour may be used instead. This gives directional lighting - a pixel placed to the **left** of centre will light up the whole circular area, but the colour will only be seen from the **right** side of the object - you can experiment.

There are two important differences between the use of the opacity texture for reflection and when used for the normal transparency effects:

- if the reflector object were made in 3DSMax/gmax using a square 2 poly plane, the opacity file will not "cut out" the circular shape of the reflector, as a circular opacity map might normally do. The black border will show in the day time. If you want the circular shape to be apparent in the day time, the object itself must be that shape; and
- the opacity map acts differently, the reverse of transparency. The white area allows the yellow light to shine through, in a circular shape, the black area blocks the yellow light.

This technique can be applied to normal scenery objects, and does not require a nightwindow option, or can be used in addition to a nightwindow directory. It could be used to make a concrete area light up at night with pools of light, if the texture is tiled. Choose your colours carefully as the effect can show in the day time.

The following image is the 3dsmax maps rollout showing the texture files as entered in the three slots (the gmax display is similar).



When exported, the texture.txt file for the plane has the Tile=st option. By changing this to Tile=s the line may become invisible. The s and the t options are related to the x and y axis, (tiling in the x or y direction) so experiment to find the correct option to delete.

The plane may have transparency on the top and sides, so use the Tile=none option to turn off the tiling for both directions.

Texture.txt file example for a separate .bmp opacity map:

```
Primary=river1.tga
Alpha=river1.bmp
Tile=none
```

Texture.txt file example for an alpha channel within the .tga file:

```
Primary=river1.tga
Alpha=river1.tga
Tile=none
```

9. Opacity on Roads, Track and Bridges

To blend a road or track model into the ground, track built-in to Trainz has used an opacity map as part of the ballast or road diffuse texture. The edges of the road or track are made transparent, blending into the opaque track or road texture.

Often a spline rail bridge is constructed with initiators and terminators. These may have a solid deck as part of the model. When using track with opacity mapping applied (Alpha channel) on a bridge, the transparency can "cut through" the textures of the bridge deck, and make it transparent when viewed from the track level.

Specify a track type that does not use transparency, such as the no ballast options available in Trainz, to fix the problem. Alternatively, model the track as part of the bridge deck, and use an invisible track as the spline for the model.

You could also use the Auran no ballast track across a truss bridge for instance, then model the track check rails as part of the bridge model.

10. Opacity Texture Bleeding

Opacity maps or Alpha channels may be used to make large areas of a plane transparent, for example making a scenery backdrop to place at the edge of the baseboard. A scenery picture is used for the texture, and the sky area above the tree line is made transparent using an opacity map as the Alpha channel. This allows the Trainz sky to show through the backdrop plane.

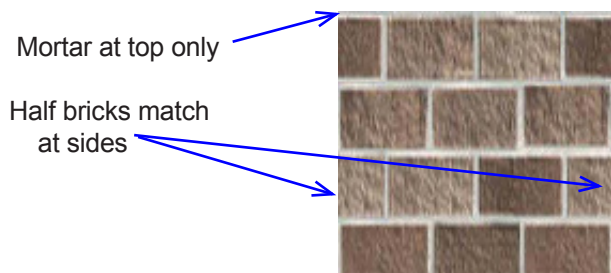
When viewed from some directions, often the top edge of the backdrop plane in the sky area shows a phantom line, being the texture colour bleeding from the bottom of the plane to the top.

If you re-export the model after changing these values, they will revert to the original Tile=st settings. You will need to amend the lines again.

11. Textures for Tiling

Tiled textures are used to cover a large surface with a small high quality picture file. The tile is mapped multiple times across a surface. The tile needs to be seamless (the patterns match and repeat at the edges) so no distinct line is visible at the joins.

For a brick pattern, the brick shape and colour should match at the sides to make full bricks, and a mortar joint is placed only at the bottom or top of the tile, otherwise a double thickness mortar line will appear at every tile join.



CREATING AN INTERIOR FOR TRS

Overview

The creation of interiors is probably the most time-consuming mesh asset you can make for Trainz. This is no doubt the reason why there are very few custom interiors around.

In fact it is quite common to spend 2 weeks of full 8 hour days on a single interior. Excluding research time, tweaking and testing. Auran uses 3dsmax and Photoshop for modeling and texture creation. One major advantage of 3dsmax over it's simpler counterpart gmax is the ability to render images. As you will read further on, rendering images has been an integral part of the realtime texture creation of Auran's released interiors.

As every interior is fundamentally different, each has had it's own issues and requirements. I.e. Steam loco's have fire, and animated levers, Electric loco's need pantograph levers (while Diesel and Steam don't), and Diesel loco's may have a Dynamic brake and other specific requirements.

Modeling and Texture Passes

We'd like to give an outline of how we went about creating the mesh, the textures and the implementation of these combined to give you a better understanding of the interior asset's structure.

- Phase 1 - Research
- Phase 2 - Modeling
- Phase 3 - Hi-res Textures and Placement
- Phase 4 - Lighting Placement
- Phase 5 - Rendering for Realtime
- Phase 6 - Realtime Textures and Placement
- Phase 7 - Realtime Model Spit-up and Attachments
- Phase 8 - Exporting and Config setup.

Phase 1 - Research

Much of the information we have found has come straight from the internet. I suppose Auran has had the luxury of making 'generic' interiors where possible by re-using the control mechanism in other interior shells.

The main focus for the generic cabs has been to make the window and side door layout correct. The general interior layout such as beams, electrical boxes, sound proofing hessian and grills has really been up to the artist to make look convincing. This is certainly the case if there isn't a lot of information available.

You can often come across a photo of the front and one side of an interior, but rarely will you find a photo of the back!

One thing to bear in mind though is that you don't have to create a perfect prototypical representation. As an artist you have a bit of 'artistic-licence' and flexibility to place and arrange things that simply add to the feel of the cab. Take the animated fan in the DD40 cab for example.

Phase 2 - Modeling

We start the model under the premise that it will be used for in-game purposes. We model it fairly low poly. Sometimes we may add some specific detail *i.e. pipes* if necessary, that will be removed after the rendering phase.

Note: Many game developers use very high polycount models for rendering realtime textures, and then assign these to low polycount models for realtime purposes.

The interior shell, the levers, accessories, windows and piping should all be modeled at this stage.

The interior polycount limits are a little more flexible in TRS due to the increased minimum spec machine requirements. However don't go silly! ☺

As an indication, here are the polycounts for a selection of Auran's realtime interior components:

QR PB15 Steam Cab Interior

- Cab shell, exterior, pipes, dial cylinders, valve bases and firebox = 7941 polygons
- Blowdown lever = 86 polygons
- Boiler needle = 14 polygons
- Brake lever = 86 polygons
- Cylinder clean lever = 40 polygons
- Animated fire panel = 513 polygons
- Animated water injector lever = 122 polygons
- Animated regulator lever = 244 polygons
- Animated reverser lever = 132 polygons
- Animated seat = 132 polygons
- Sander lever = 28 polygons
- Whistle lever = 68 polygons

SBB Krokodil Electric Cab Interior (available for download)

- Cab shell, dash board, seats, gauges, bonnet and fire extinguisher = 3362 polygons
- Brake wheel = 220 polygons
- Horn lever = 46 polygons
- Reverser lever = 124 polygons
- Throttle wheel = 154 polygons

SNCF TGV Electric Cab Interior

- Cab shell, dash board, seats and windows = 1886 polygons
- Brake lever = 54 polygons
- Pantograph lever = 94 polygons
- Throttle ring = 96 polygons
- Reverser lever = 38 polygons

UP DD40 Diesel Cab Interior

- Cab shell, exterior, dash panel, windows, brake lever bases, handrails, driver seat and fire extinguisher = 2629 polygons
- Animated fan (including blades) = 354 polygons
- Swivel seat = 180 polygons
- Sun visor = 94 polygons
- Animated wipers = 96 polygons
- Train brake lever = 60 polygons
- Loco brake lever = 50 polygons

Phase 3 - Hi-res Textures and Placement

This is the first of three texture passes undertaken to create realtime textures.

We map hi-res textures to every surface of the model using the mapping commands from the 'Modify' panel of 3dsmax - 'UVW map' and 'Unwrap UVW'.

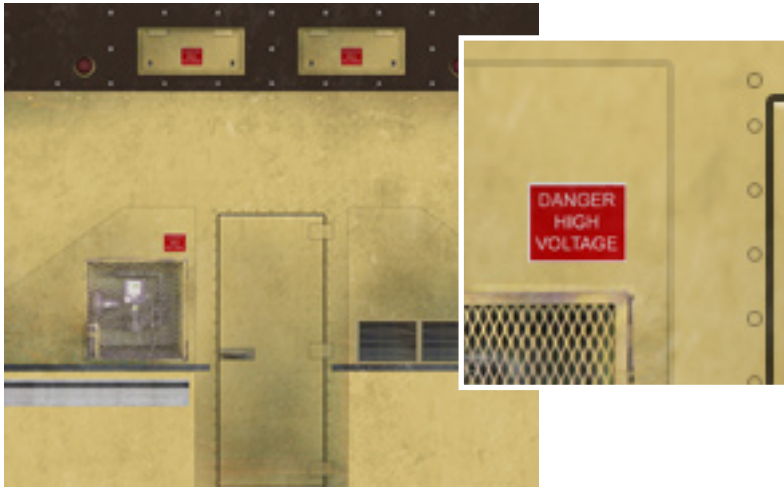
Below are examples of hi-res textures assigned to the first pass of the DD40 interior front. Generally match the ratio of physical model dimensions to the texture. This way we eliminate details being stretched when mapping.

It is fairly important to limit the highlight and shadow variations to the hires diffuse textures as this will be created by 3dsmax during the render phase. Adding dirt and subtle surface texture is usually necessary (keep detail and texture consistent over all hires textures).

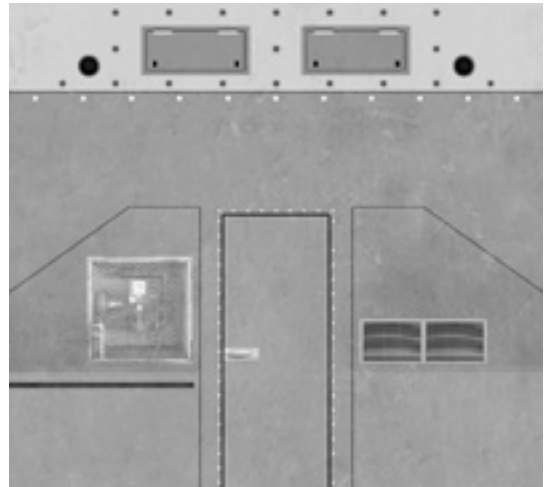
In many cases we will also create and assign greyscale bump maps to add additional surface variation and relief.

You can alter the material properties by altering the 'specular' and 'glossiness' settings before the next texture pass of rendering.

Hi-res diffuse map (800h x 887w) for the DD40 cab



Hi-res bump map (800h x 887w) for the DD40 cab



Phase 4 - Lighting Placement

Once the first texture pass is finished it is now time to strategically add lighting to the scene. 3D Studio Max ships with a variety of light types.

The ones we generally use are the following:



Target Spotlight

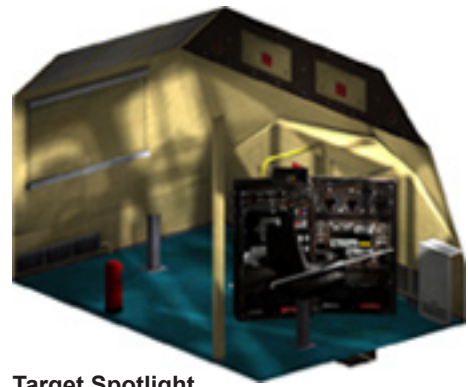
The main light source for the scene.
Acts as the 'sun'.



Omni Light

Ambient lighting and additional highlights.
- given a negative 'multiplier' value
removes light to accentuate shadows.
I.e. Under benches.

You'll have to experiment with lighting
and add variation to make it look right.



Target Spotlight

Multiplier: 1.6
Contrast: 0.0
Soften Dif. Edge: 100.0
Far attenuation: Start: 9.0m
End: 14.0m

Decay: none
Object shadows: On
- Shadow map

Omni light

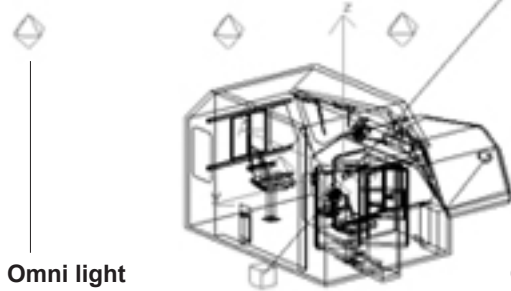
Multiplier: 0.6
Contrast: 100
Soften Dif. Edge: 100.0
Far attenuation: Start: 5.0m
End: 10.0m

Decay: none
Object shadows: On
- Shadow map

Omni light

Multiplier: 0.4
Contrast: 10
Soften Dif. Edge: 100.0
Far attenuation: Start: 2.0m
End: 3.0m

Decay: none
Object shadows: On
- Shadow map



Omni light

Multiplier: 0.7
Contrast: 50
Soften Dif. Edge: 100.0
Far attenuation: Start: 2.0m
End: 3.0m

Decay: none
Object shadows: On
- Shadow map

Omni light

Multiplier: 0.7
Contrast: 70
Soften Dif. Edge: 100.0
Far attenuation: Start: 2.0m
End: 3.0m

Decay: none
Object shadows: On
- Shadow map

Phase 5 - Rendering for Realtime

The next step is to the second texture pass. This involves rendering all details and surfaces (front, back, sides, floor, ceiling, door recesses etc.).

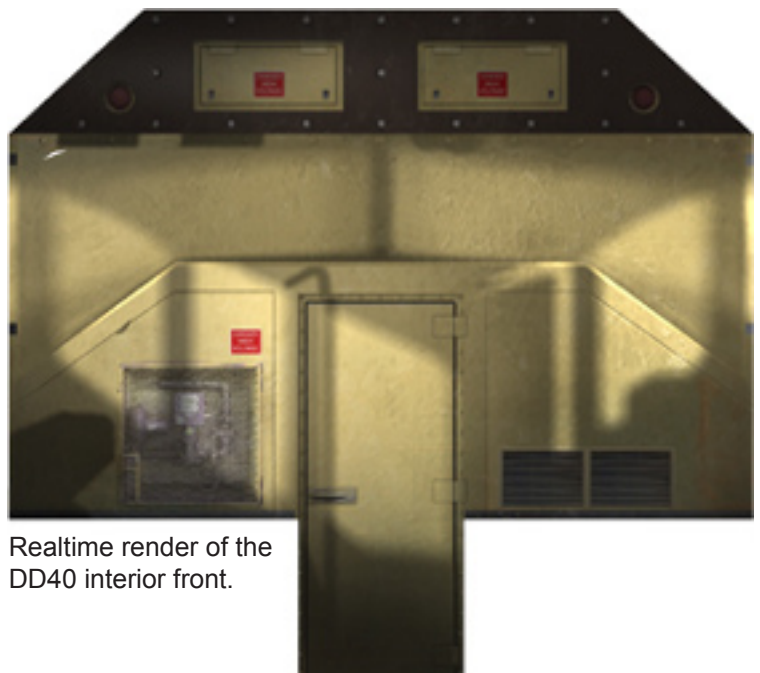
There is a very handy 3dsmax material plugin called "Cast Shadows Only" from Blur Studios. This material makes the object invisible, but lets it cast shadows. Very handy when trying to control what does and doesn't render in a scene.

We create another cab shell with the window cut-outs to completely surround the textured cab shell. This casts the window shadows over the model.

Download the 'Cast Shadows Only' plugin
<http://www.3dmaxer.dk/plugsdown4.asp?id=45>

This plugin requires a utility called 'blurlib' available for download:

<http://www.3dmaxer.dk/plugsdown4.asp?id=43>



Realtime render of the
DD40 interior front.

Phase 6 - Realtime Textures and Placement

This Phase takes all the rendered out textures (from Phase 5) and manipulate them into a format that TRS can read.

The following example is one of the realtime textures from the DD40 interior. Note the front render described before is now part of the realtime texture. As are the two sides and the rear render.

The front door however has been rotated 90 degrees and the floor render is in its place. This is in order to economically utilise the texture space.

Once the render manipulation is complete, you will probably have to re-assign much of the new textures back onto the model. In many cases this should just consist of re-doing the Unwrap UVW command.

Phase 7 - Realtime Model Spit-up and Attachments

After all your realtime textures have been mapped, it is now time to split up the model into logical divisions and add attachment points.

Doing this in the logical order described below will save a lot of time and frustration.

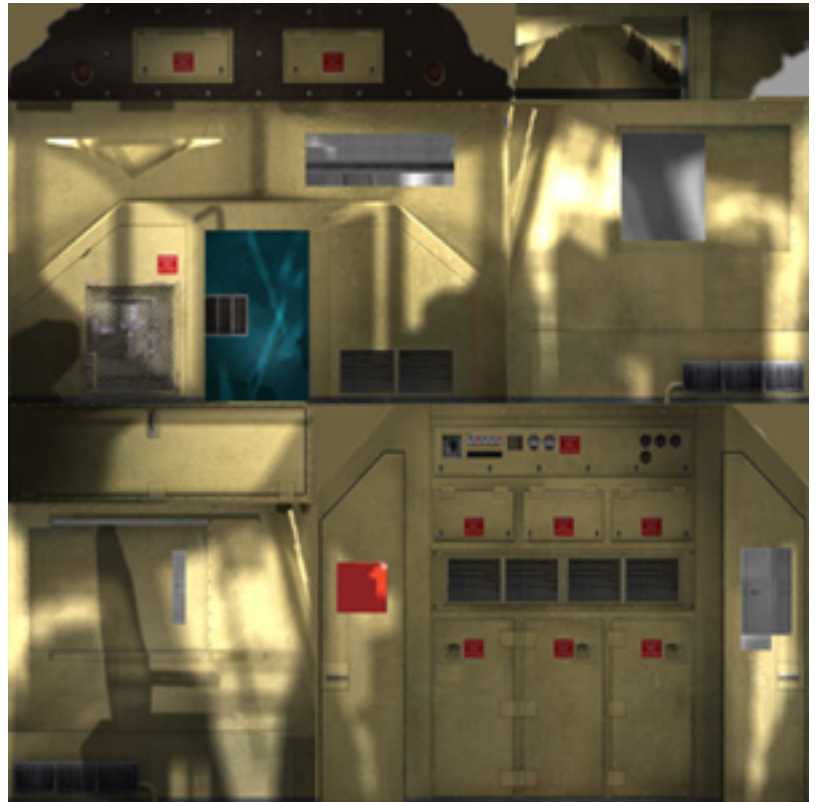
Remember to save a non-split version for backup!

Any object in the scene that has to move (ie. levers and dials), or animate, will need to be 'detached' from the main model. To do this, select the polygons or elements you want detached and through the 3dsmax modify panel, press the *detach* button and name the new object.

To make life a heck of a lot easier later on, it pays to move the *pivot point* of each new object to it's logical position. Ie. For needles and levers the pivot point should be the centre of rotation (with the rotation around the Z axis). Align lever shaft in the Positive Y direction.

To do this, access the 3dsmax *Hierarchy* panel, and press *Affect Pivot Only*. From here you can centre and align the pivot to the object and move it to the preferred

Realtime texture example of the DD40 interior.



location.

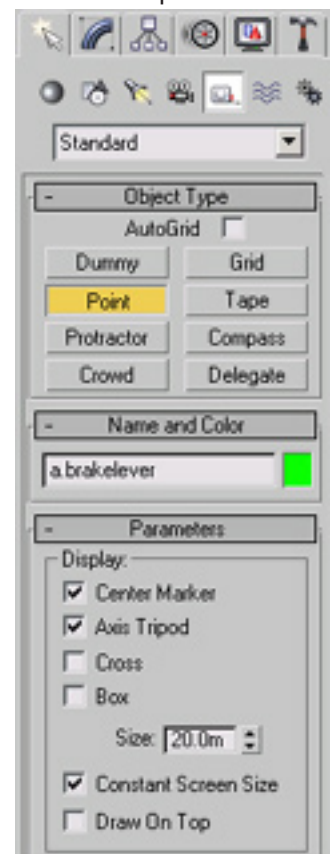
Once the pivot points have been set for all your new objects, now is the time to start adding attachment points. Add these through the 3dsmax *create* panel (see image below). Remember to use the *a.name* naming convention. Attachment points are simply points in 3D space where another mesh can be inserted (through config.txt reference).



Since you have already worked out the pivot points of all your objects, all you have to do is align the attachment points to your object pivot points. Press the icon (above) and select the object to align to....

Align position: X, Y & Z
Current and Target: Pivot Point

Attachment point creation



Note: gmax users...

It is of course possible to skip Phase 3 to 6 and create the realtime textures yourselves with a 2d editing program such as Photoshop or Paintshop Pro before assigning them to the model.

Study the cab interiors that Auran has included in TRS as correct lighting variations truly add to the 3D feel of the interior.

Studying how light falls on and reflects off different materials and adding this knowledge to your textures could be the difference between a nice cab and an awesome one.

3dsmax does remove a lot of the guesswork and has the ability to create flowing, seemingly natural lit scenes. It might take a little extra time but the results are worth it! Rendering options are not available in gmax.

These attachments are exported as part of the default mesh (or the main mesh). Objects for attachment must not be included in the export as they are each exported from their own 3dsmax files and attached to the default mesh separately (through config.txt reference).

Save a backup of the 3dsmax file!!

Select each object from the scene and save each one as its own 3dsmax file, (command panel, *file - save selected*). You may delete the object from the main mesh scene after you save it.



The next step is to open each 3dsmax file you have just created (of the movable objects), and move the object to 0,0,0 and make the rotational values 0,0,0 (press and right click these 3dsmax icons). This makes the object (and it's pivot) aligned correctly to the origin of the scene.

Now is the time to export the object to a Trainz mesh file (.im).

Note: When a lever is inserted into the default mesh it is attached at the lever's origin, regardless of the geometry the lever has. Setting the pivot point of the object earlier, and aligning the object to the origin in it's own scene, reduces the possibility of alignment errors.

The rotation is always through the attachment point's Z axis and the in-game notch display defaults from the positive Y location (clockwise) around the Z axis. Altering the angle settings in the config can adjust the notch position.

Refer to the config.txt example earlier

Phase 8 - Exporting and Config setup.

Exporting the objects is very simple. You will need the Trainz Exporter Plugin for 3dsmax, http://www.auran.com/TRS2004/downloads/contentcreation/TRS_Max4_Plugin_Bump.zip

or the gmax Asset Creation Studio, <http://www.auran.com/trainz/contentcreation.htm>

Select the objects within the scene that you want exported, from the command panel, press *File, Export Selected*, select Trainz Format and remember to type in the file extension in the File name dialogue box.

All mesh files should be within the same directory as the config (or in a directory within the same path). The config.txt may contain a sub-path to find the mesh, e.g.

When exporting the default mesh, you must include all attachment points. TRS may crash if the config.txt references an attachment that is not there. *Config.txt file set-up*

There are a few key things to remember.

1) Mesh-table:

An interior uses the same properties as any TRS mesh-table, you need to add `auto-create 1` or the mesh will not show in the scene.

There may be occasions where you *don't* want a mesh visible by default. Take the switchlights for example. These have `auto-create 0` in the config as their visibility is controlled by script when the switch is in the on position.

2) Animations:

Note the animated fan and the wipers do not have the `animation-loop-speed` tag added. This is because the animations are controlled through the script. They are visible by default, but the animation does not play by default.

Should we have added `animation-loop-speed 1` the looping animation would have played by default (with or without a script)

3) Levers:

The visors, swivel chair and sliding windows are all setup as levers.

In these three cases the notchheights are 0 so they don't display.

In the case of the visor and sliding windows, they each have a number of invisible notches. This is to give the user the option to have them 'slightly' open or 'partially' closed. ☺

The sliding windows have a very large radius (30m), and very small angles.

STEAM CAB INTERIORS

Overview

TRS steam cab interiors have been set-up in generally the same way as diesel and electric cabs with a few additional steam specific features.

Many of the levers and fireplates have several moving objects and required mouse controlled animations. This differed from the usual lever types with only one object, set to rotate around an attachment point.

Not only did the levers need reviewing, but the cab firebox itself had to produce fire and glow variations and the coal shoveller needed to be controlled and linked to the coal requirements also.

Download PB15 Interior source and in-games files here: http://www.auran.com/TRS2004/downloads/contentcreation/TRS2004_PB15_interior.zip

Animated Levers

Animated levers are generally set up like all other animations in TRS/Trainz. Bones (or dummies) need to come with the b.r.name naming convention.

As the new animated levers are mouse controlled, the need arose to be able to 'grab' the lever handle only and not the rest of the animated parts. Because of this, animated levers require a collision mesh (kind: collision-proxy)

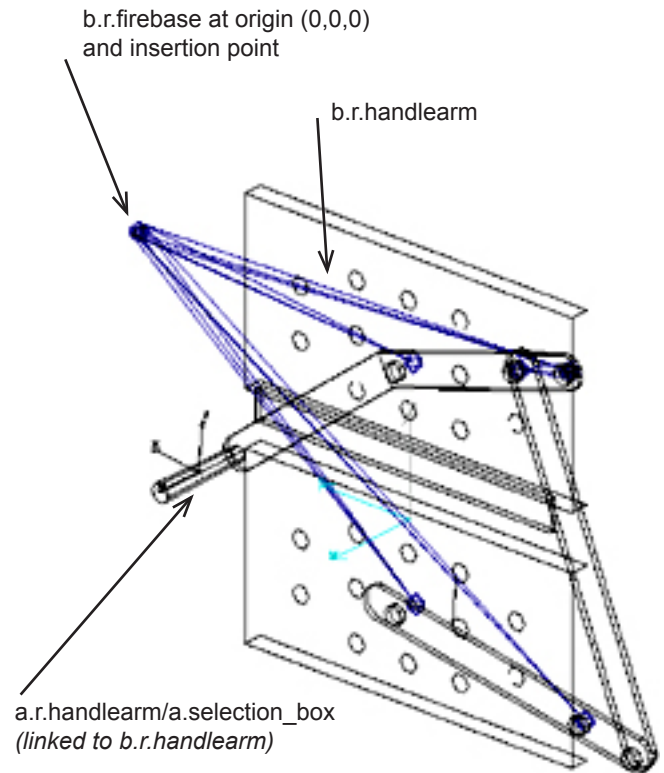
Take the animated fire plates for example:

```
fire_plates
{
    mesh fireplates/fireplates.im
    anim fireplates/fireplates.kin
    auto-create 1
    kind animated-lever
    test-collisions 0
    notches 0, 1.0
    notchheight 1,1
    limits 0, 1.0
}
fire_plates-collision-box
{
    mesh fireplates/selection_box/selection_box.im
    att-parent fire_plates
    att a.selection_box
    auto-create 1
    kind collision-proxy
    opacity 0
    collision-parent fire_plates
}
```

The fire_plates are kind animated-lever. The mouse cannot select this mesh as it has the test-collisions 0 tag.

Note the fire_plates-collision-box has auto-create 1 but has an opacity 0. Also, the parent mesh it defaults to is the fire_plates. That is, you have to mouse over the fire_plates-collision-box in order to move the fire_plates.

The a.selection_box attachment is named **a.r.handlearm/a.selection_box** in 3dsmax as it is 'linked' to the animated bone called **b.r.handlearm**.



Note: Moving attachment points for other models.

Attachments use the a.name convention and allow the attachment of a submesh to a specific point in a mesh. If the attachment point in the main mesh is to move with an animation of that mesh, the special naming convention above must be used for the point to follow the animation, and allow the submesh to then follow the point.

For example, a submesh is to be attached to a main mesh using the attachment name a.lever. This attachment point in the main mesh must be linked to a helper point called b.r.pivot, that is animated.

The attachment point in the 3dsmax/gmax model must be named a.r.pivot/a.lever. In the config.txt file the attachment point will be entered as a.lever. The submesh will now follow any animation in the main mesh. It can be useful for moving coronas, and animated nightwindow meshes.

Steam Cab Fire and Coal Glow Effects

These effects are generated automatically by TRS when it finds *firebox*, *fire*, *coal*, and *fireglow* in the config.txt.

```
firebox
{
  mesh firebox.im
  auto-create 1
  kind firebox
  light 0
  test-collisions 0
}
fire
{
  mesh fire.im
  auto-create 1
  light 0
  test-collisions 0
}
coal
{
  mesh coal.im
  auto-create 1
  light 0
  test-collisions 0
}
fireglow
{
  mesh fireglow.im
  auto-create 1
  light 0
  test-collisions 0
}
```

Note the tag: light 0. This is because the mesh's lighting is dealt with differently through code to resemble the gradual glowing of the coal and fire heating up.

Firebox

This is simply a textured box. TRS controls lighting effects when the temperature rises

Fire

*This is simply a few polygons with simple **planar UVW mapping** with a texture called *flametest2.tga*. The UVW mapping is altered automatically by TRS to play each frame of the fire animation.*

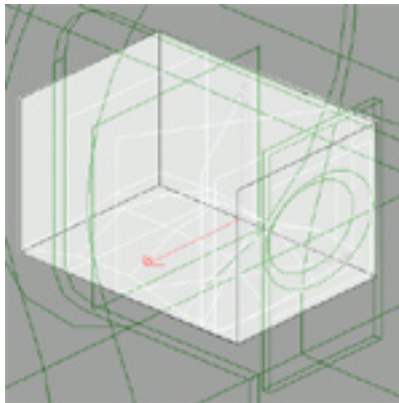
coal

*This is simple mesh with simple **planar UVW mapping** with a texture called 'coalfire.tga'. The UVW mapping is altered automatically by TRS as the temperature rises.*

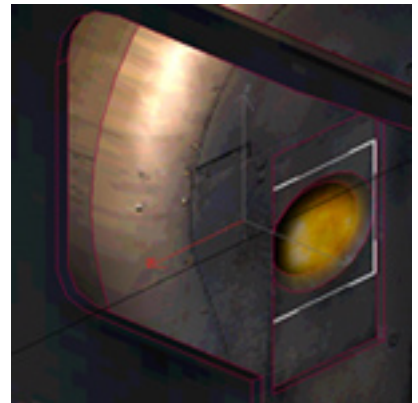
fireglow

This is single polygon that acts as a visible glow around the firebox opening. Visibility is controlled by TRS.

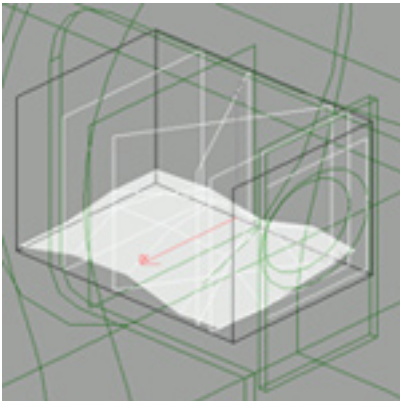
firebox mesh



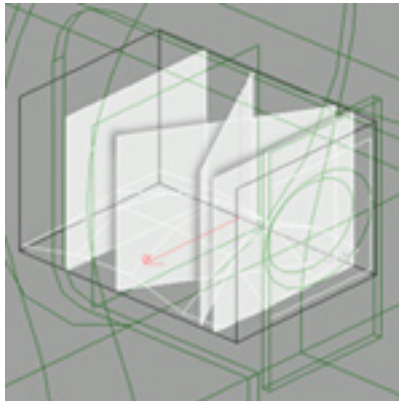
fireglow mesh



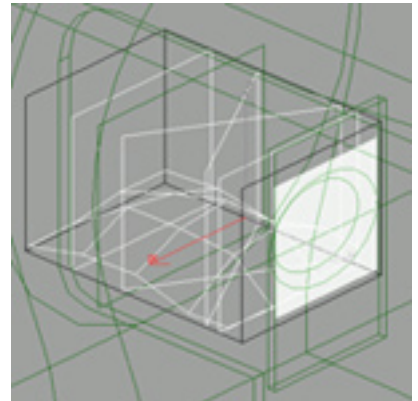
coal mesh



fire mesh



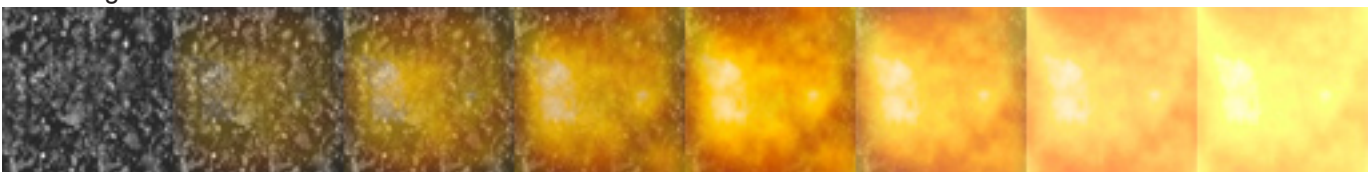
fireglow mesh



flametest2.tga



coalfire.tga



Researching Data and Testing of a Steam Locomotive



Steam locomotives are quite complicated and many of the performance values entered as tags in the config.txt file interact with, and influence, each other. This can make it difficult to firstly choose starting values for the various input data, then to vary those values during testing within Trainz to produce a smoothly operating and realistically performing locomotive.

Using the Steam Locomotive information in the Content Creation Guide and entering data for the various settings, the following may assist in developing a workable and realistic steam locomotive.

Research and Record:

1. the locomotive data and convert the data to metric values (the config.txt is ALL in metric);
2. the dimensions for the cylinders, bore & stroke;
3. boiler working pressure;
4. the specific locomotive hauling rating on what percentage grade (for cut off and speed);
5. the normal service speed with that load on the level, and its practical maximum speed; and
6. the water and coal consumption per hour (sometimes the hardest to find).

Input in the Config.txt file:

1. *piston-area* in sq metres;
2. *piston-volume-max* in litres;
3. *piston-volume-min* of the cylinder at 3% of the maximum volume;
4. set the *initial-boiler-temperature* at 80 - 85% of working boiler temperature;

5. set the safety valve values (see comment below);
6. suggest *firebox-to-boiler-heat-flow*, and *burn-rate*, of 0.06 as a good starting point;
7. *boiler-to-piston-flow* and *piston-to-atmosphere-flow* usually 0.0035;
8. the *water-injector-rate* at 15 to start, *shovel-coal-mass* at 35, and *fuel-energy* value at 20;
9. the *boiler-volume* approximately 10x real volume (see the table in the Content Creation Guide for three sets of figures that cover "proportionally" appropriate loco sizes).

Testing Suggestions:

1. starting with the loco and train on a long level section adjust the speed and power with the boiler to piston and piston to atmosphere flows;
2. carefully adjust the "firebox-to-boiler-heat-flow", the "max-fire-temperature" along with the "fuel-energy" settings (the boiler steam production can be made to be prototypical and produce enough steam providing the locomotive is hauling a train within its capacity and is driven with the correct use of the valve gear);
3. during this process, the injector is tuned in to feed enough water at just below maximum demand;
4. the shovel rate is used to adjust the amount of coal used and to supply heat;
5. the fuel value is another adjustable in the fire parameters;

This should give a config.txt file that reasonably reproduces the real performance in Trainz, hauling standard Auran cars (most users don't vary or adjust the rollingstock config.txt files, yet it is not difficult to find tonnage and speed numbers for the locomotives).

A comment on the safety valve settings: Try not to waste water through the safeties, keeping the settings close together (by setting the safety valves to 15Kpa (2lbs) difference) allows for a rapid release to bring the pressure back quickly, thus emulating the action of "Pop" type valves. This is both deliberate and prototypically accurate, to prevent over pressure and ultimately the big bang. More practically in Trainz, it prevents water wastage.

Narrow Gauge Geared Locomotives

These engines have relatively small cylinders and they use a miniscule amount of water. To get the water usage up to an appropriate prototypical rate, you will need to multiply the cylinder volume by the gear ratio then you will get it to use the appropriate amount of water.

The cylinder volume makes no difference to the performance, just the water usage, so for a geared locomotive such as a Shay, Climax, or Heisler, by multiplying the cylinder volume by the gear ratio then the

amount of water used will be realistic. It is the piston area / boiler pressure that gives the output force / power that drives the locomotive and train.

For example, a typical Shay locomotive may have a gear ratio quoted as 3.3:1 (3.3 to 1). Multiply the boiler volume by 3.3 to give a realistic water usage.

Tenders

Locomotives with tenders normally operate as a unit. However, if you use the runaround comand in Driver the locomotive leaves the tender behind.

For steam tender config.txt files, add the following line:

tender 1

This ensures the tender stays with the locomotive.

Tenders Dump Coal

To prevent steam tenders or the coal bunker of a steamer being robbed at the Power station or any Multi industry track that is coal unload enabled, you should use the “No Dump” Script by Wulf_9.

You may find the scripts in the following forum post. Please read the post carefully.

<http://forums.auran.com/TRS2004/forum/showthread.php?s=67dd789e0493d937324e5dda2c17090a&threadid=62630>

Include the script with the asset and add these lines to a tender config.txt file and it stops the tender dumping coal incorrectly.

```
script “steamtender”  
class “Steam_Tender”
```

A second script is available for tank engines, and the following lines should be included in the config.txt file for the engine:

```
script “steamtank”  
class “Steam_Tank”
```

Content Creation Guide Amendments

This section lists comments on, and additional changes to, the Guide that have been made since version 1.04 was released. They may be incorporated in future updates of the Guide.

Ditch lights

While ditch light had been mentioned in the guide these have not been fully implemented (tested) and reference has been removed.

In TRS, ditch lights may be simulated using the corona attachment method mentioned in the Guide. This however does not provide any surrounding illumination to the track.

Naming of Assets

Various asset kinds are automatically named. This enables the name to be changed by clicking on the “?” key in the Surveyor menu then on the placed asset.

For other assets that do not display this automatic function, you can also specify that you want to enable auomatic naming by adding the following line to your config.txt file:

autoname 1

ANIMATION EVENTS

Sounds events and generic events can be linked to an animation key-frame to give great control over sound and script timing for industry and scenery assets.

When an animation file (.kin file) is exported from 3dsmax or gmax, the exporter will make a query for an event file. Tick the box and you will be asked to browse to the event file. See *Figure 1*. The event information is added to the contents of the new animation file.

The Event File: (.evt)

Format: FrameNum EventType EventName.

The event file consists of a list of events and is set up as a simple text file. Each event consists of the frame number, followed by the event type (Sound_Event or Generic_Event), then the event name. Sound events are generally referenced as a trigger within the asset's config.txt file (or through script). All events start on a new line (should there be more than one)

Sound and Generic Events:

A Sound_Event tells TRS when to play a sound, relative to an animation keyframe.

A Generic_Event is an animation keyframe reference for script timing and control.

Example 1: Lumbermill:

The animation in the max file is set up over 1000 frames.* We want a single sound to play on frame 760 as a log runs through the mill on the conveyor. When exporting the kin animation we are queried for an event file.

lumbermill.evt

```
760 Sound_Event logcut
999 Generic_Event animstop
```

An excerpt from the TRS Lumbermill config.txt file (note the trigger) :

```
soundscript
{
    log_cut
    {
        trigger logcut
        attachment a.sawsound
        nostartdelay 1
        repeat-delay 1
        distance 10,400
        sound
        {
            log_cutting.wav
        }
    }
}
```

Note: For script reference please refer to index.chm found in \Trainz\scripts\docs directory.

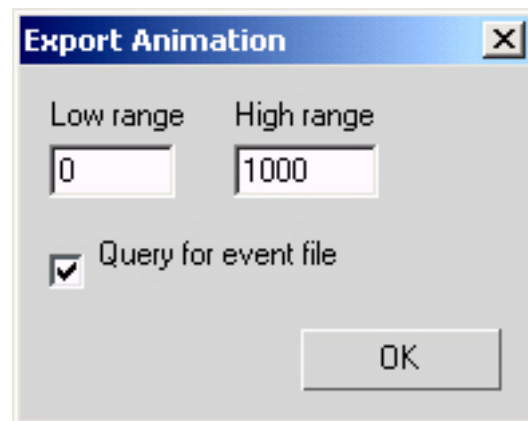


Figure 1. Export Animation window.

Example 2: Looping sounds:

Not only can we control when a single sound plays but we can also control the start and the stop of a looping sound by adding another Sound_Event with a / before the relevant name. In the example below the sound starts on frame 370 and ends on 589.

typicallooping.evt

```
370 Sound_Event reverse
589 Sound_Event /reverse
```

```
soundscript
{
    backup
    {
        attachment a.sound
        trigger reverse
        repeat-delay 0
        distance 5,100
        sound
        {
            warning.wav
        }
    }
}
```

Important Note:

The Max file in Example 1 has 1000 frames:

One thing to note is that although the Max file states the frames are from 0 - 1000 frames, we must remember that frame 999 is the last one.

On a looping time scale, frame 1000 is the same time as frame 1.

Therefore, the Generic-Event at the end of the animation is thus:
999 Generic_Event animstop

LEVEL OF DETAIL MESH REDUCTION

General Description

Level of Detail (or 'LOD') is a technique used for asset mesh reduction. Trainz uses a different mesh dependant on the viewing distance.

This concept is different from the previous "progressive mesh" (.pm) reduction as used by UTC. That is, instead of gradually reducing the polycount of a single mesh you can now have several versions of the same asset, each at different polycounts and texture levels. (See 'Directory Structure', next page)

Assets with LOD reduction must comprise of 'indexed meshes' or .im files only (exported from gmax or 3dsmax). No .pm files are used in LOD.

TRS2004 looks for these .im files through an .lm.txt (LOD mesh file) which is referenced via the asset's config.txt file.

Note: Only Figure 4 the hi-res version is bump-mapped. Bump mapping will be ignored if the graphics card does not support it.

Use only non-formatted text to create the .lm.txt file i.e.
Use a simple text editor such as notepad

Refer to the .lm file in the next column for information.

LOD Mesh File (PB_15_body.lm.txt) Auran's steam loco:

```
version 1.0
offset = 0.01;
calcPoint = center;
multiplier = 1.0;
animationCutoff = 0.00;
renderCutoff = 0.00;
attachmentCutoff = 0.06;

mesh ("0.07")
{
    name="PB_15_body_lowest.im";
}

mesh ("0.30")
{
    name="PB_15_body_low.im";
}

mesh ("0.52")
{
    name="PB_15_body_med.im";
}

mesh ("1.0")
{
    name="PB_15_body.im";
}
```

Breakdown of LOD Mesh File

Version 1.0

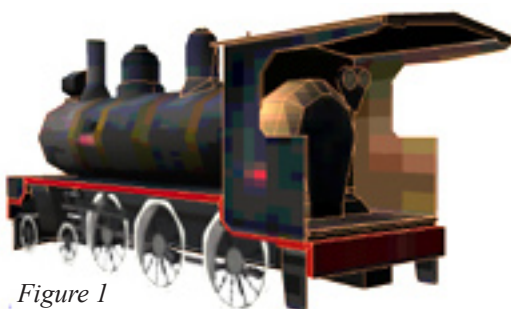
offset = 0.01;

The offset that prevents "popping" between two levels of detail- repeatedly.

calcPoint = center;

The position where the level of detail is calculated from (center,near,far)

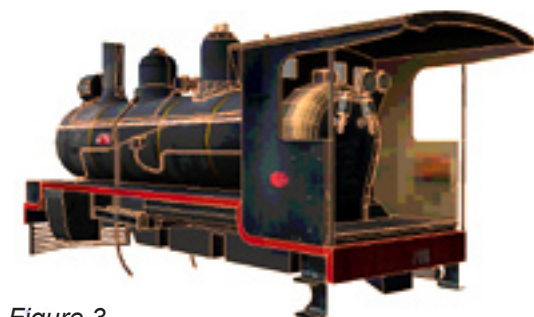
PB_15_body_lowest.im (600 polys, 64x64 tex) bogies attachments flagged 'Cull', bogies represented in mesh (see next page)



PB_15_body_low.im (1947 polys 256x256 tex)



PB_15_body_med.im (5066 polys 512x256 tex)



PB_15_body.im (10578 polys, 1024x512 tex & bump-mapped)



multiplier = 1.0;

A level of detail multiplier (leave as 1.0)

animationCutOff = 0.00;

*The level of detail where animation stops
(to screen width) 1.00 = full width,
0.5 = half screen width, 0.00 = never stop animation.*

renderCutOff = 0.00;

The level (to screen width) where rendering stops (no longer visible).

attachmentCutOff = 0.06;

*The level where **:Cull** flagged attachments are dropped.
(to screen width) * See note below*

Note: Below. Meshes referenced within an LOD file must be in ascending screen width order.

mesh("0.07")

```
{  
  name="PB_15_body_lowest.im";  
}
```

When the mesh is displayed at 0.07 of the screen, the mesh 'PB_15_body_lowest.im' is displayed. Note the figure is just bigger than the attachment-CutOff figure above. This ensures the modeled bogeys in this LOD mesh are rendered before the actual bogeys are culled.

mesh("0.30")

```
{  
  name="PB_15_body_low.im";  
}
```

Mesh 'PB_15_body_low.im' is displayed when the mesh is displayed at 0.3 of the screen.

mesh("0.52")

```
{  
  name="PB_15_body_med.im";  
}
```

mesh("1.0")

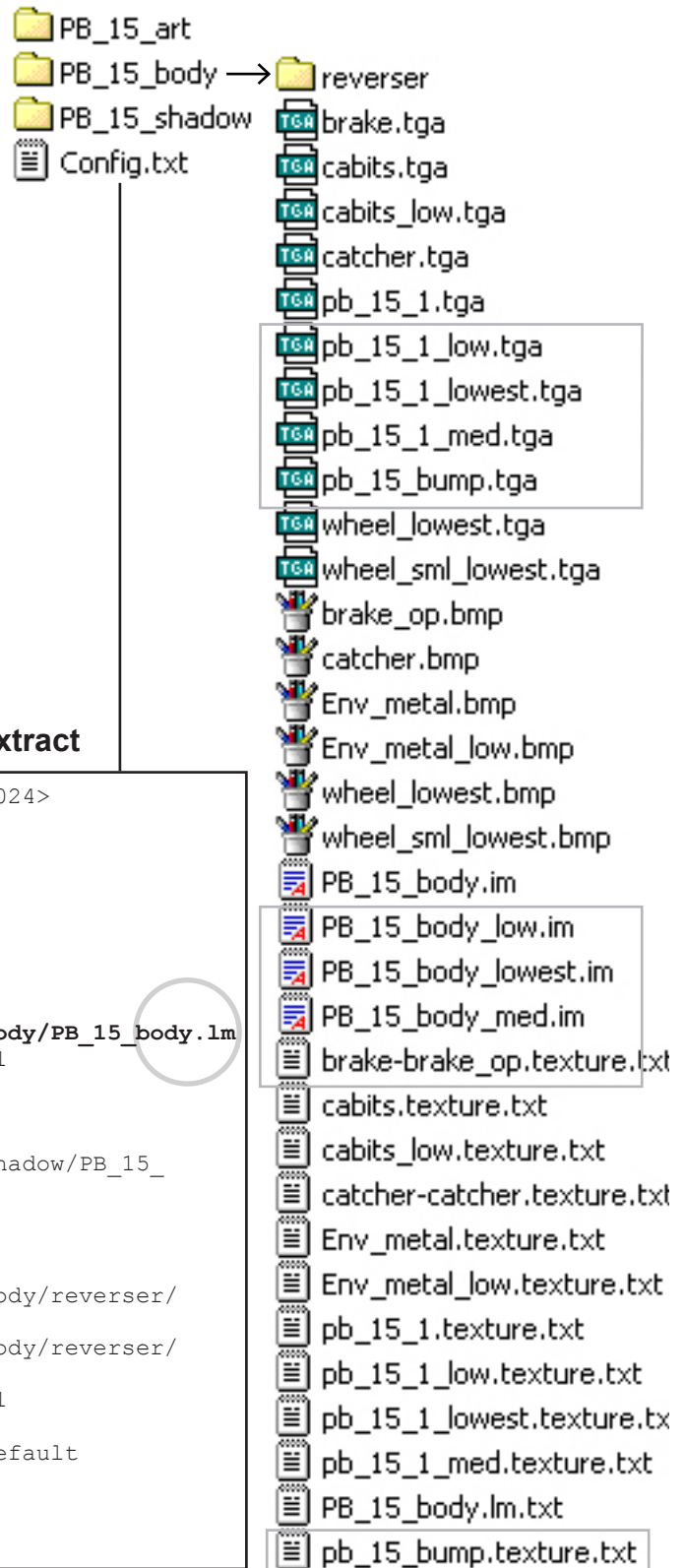
```
{  
  name="PB_15_body.im";  
}
```

Directory Structure (LOD Loco)

The main thing to remember is that all LOD files, .im meshes and textures must be located within the same directory. In the case of a locomotive or rolling stock item, it should be in the *_body directory. Note the LOD Mesh file is referenced from the config.txt file.

Config.txt file extract

```
kuid <KUID:-3:10024>  
kuid-table {  
}  
obsolete-table {  
}  
mesh-table  
{  
  default  
  {  
    mesh PB_15_body/PB_15_body.lm  
    auto-create 1  
  }  
  shadow  
  {  
    mesh PB_15_shadow/PB_15_shadow.im  
  }  
  reverser  
  {  
    mesh PB_15_body/reverser/reverser.im  
    anim PB_15_body/reverser/reverser.kin  
    auto-create 1  
    att a.bog2  
    att-parent default  
  }  
}
```



Note

*attachmentCutOff = 0.1;
Attachment cutoff specifies the level where attachments with the flag **:Cull** are dropped.*

*(ie To stop drawing the bogeys of the trains at a specific level of detail, append **:Cull** to the bogie attachment point. (ie "a.bog0:Cull"))*

Where the above applies (bogeys culled) the body mesh will need a low poly representation of the bogeys.

LOAD TEXTURE REPLACEMENT

This feature was created for rolling stock items that use animated bulk loads.

If a rolling stock item (say a Gondola) is setup to carry any product within product-category 'Bulk Load', AND it's config.txt file is set up to enable texture replacement, then a texture replacement on the load mesh will take place to visually represent the product.

Lets break down the texture-replacement setup:

Texture swapping functionality relies on 2 things:

- 1) The rolling stock item config.txt to include with 'product-texture' load effects setup (see Rollingstock examples 1 and 2 below).
- 2) The product config.txt itself needs product-texture information and reference to the texture to be loaded (see product examples on page 3).

ROLLINGSTOCK EXAMPLE 1:

COAL HOPPER

Load allows texture replacement:

Note the texture to be replaced (load_map) (see load, effects, product-texture in config.txt)

Default product load:

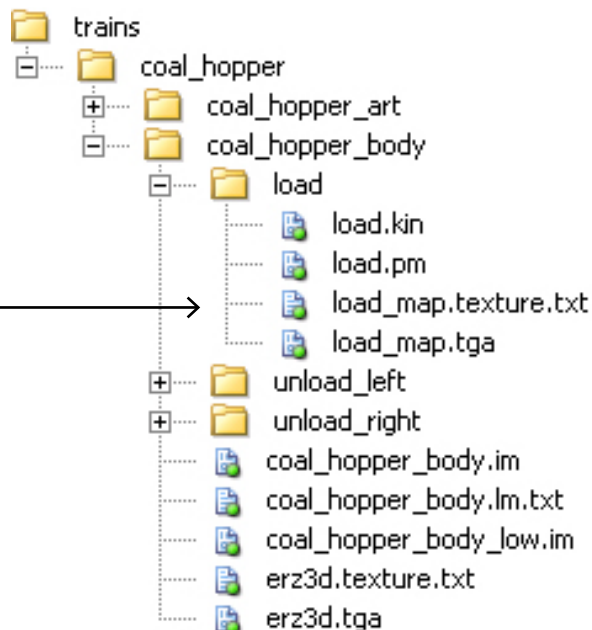
COAL <KUID:44179:60013>
(see product-kuid field in config.txt)

Can take product-category:

BULK LOAD <KUID:-3:10040>
(see allowed-categories field in config.txt)

When the Coal hopper enters an industry asset that produces a bulk-load other than it's default (ie. woodchips at a lumbermill), and it loads this product, texture-replacement will take place.

Coal Hopper Directory Structure



Extract from Coal Hopper Config.txt

```
mesh-table
{
  default
  {
    mesh coal_hopper_body/coal_hopper_body.lm
    auto-create 1
  }
  shadow
  {
    mesh coal_hopper_shadow/coal_hopper_shadow.pm
  }
  load
  {
    mesh coal_hopper_body/load/load.pm
    anim coal_hopper_body/load/load.kin
    auto-create 1
    use-parent-bounds 1
    effects
    {
      product-texture
      {
        kind texture-replacement
        texture "load_map.texture"
      }
    }
  }
}
queues
{
  load0
  {
    size 54300
    initial-count 0
    animated-mesh load
    product-kuid <KUID:44179:60013>
    allowed-categories
    {
      0 <KUID:-3:10040>
    }
  }
}
```


ROLLINGSTOCK EXAMPLE 2:

WOODCHIP GONDOLA

Load allows texture replacement. Note the texture to be replaced (load_map)
(see *load*, *effects*, *product-texture* in *config.txt*)

Default product load:

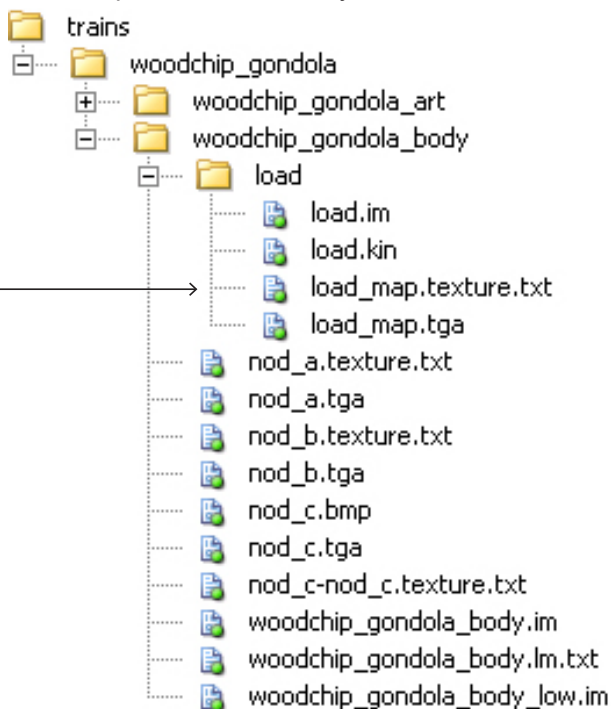
WOODCHIPS <KUID:-3:10002>
(see *product-kuid* field in *config.txt*)

Can take product-category:

BULK LOAD <KUID:-3:10040>
(see *allowed-categories* field in *config.txt*)

When the Woodchip Gondola enters an industry asset that produces a bulk-load other than it's default (ie. coal at a coalmine), and it loads this product, texture-replacement will take place.

Woodchip Gondola Directory Structure



```
mesh-table
{
  default
  {
    mesh woodchip_gondola_body/woodchip_gondola_body.lm
    auto-create 1
  }

  shadow
  {
    mesh woodchip_gondola_shadow/woodchip_gondola_shadow.im
  }

  load
  {
    mesh woodchip_gondola_body/load/load.im
    anim woodchip_gondola_body/load/load.kin
    auto-create 1
    use-parent-bounds 1
    effects
    {
      product-texture
      {
        kind texture-replacement
        texture "load_map.texture"
      }
    }
  }

  queues
  {
    load0
    {
      size 60500
      initial-count 0
      animated-mesh load
      product-kuid <KUID:-3:10002>
      allowed-categories
      {
        0 <KUID:-3:10040>
      }
    }
  }
}
```

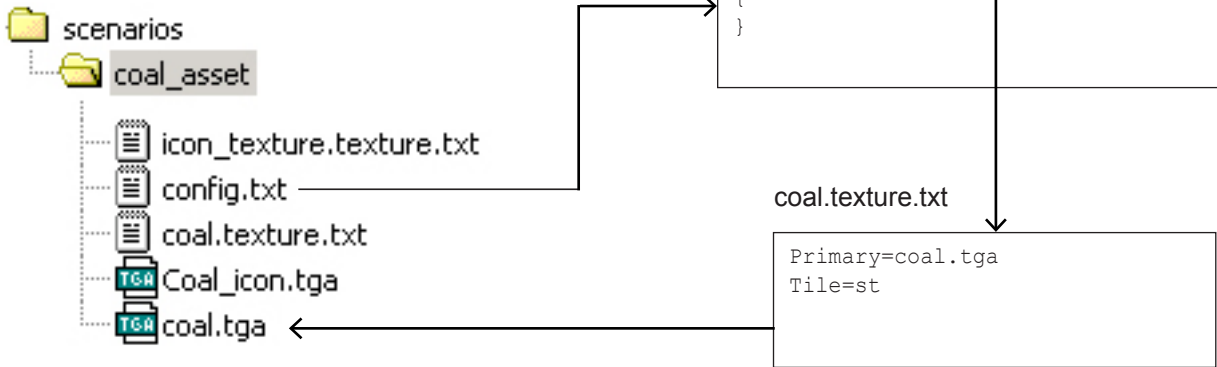
As a rolling stock item with texture-replacement enabled has reference to the texture to be replaced, the product has reference to the texture that will be used in its place.

PRODUCT EXAMPLE 1:

COAL PRODUCT

In the case of the coal product, the texture to be used is 'coal.tga', i.e. if the woodchip gondola enters the coal mine to load coal, the load-map texture will be replaced with coal.tga.

Coal Product Directory Structure



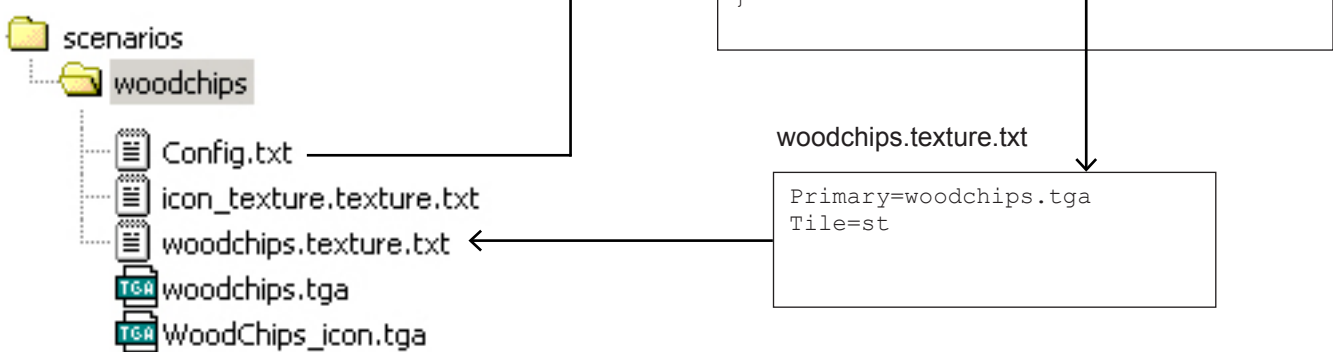
PRODUCT EXAMPLE 2:

WOODCHIP PRODUCT

In the case of the woodchip product, the texture to be used is 'woodchips.tga'

I.e. If the coal hopper enters the lumber mill to load woodchips, the load-map texture will be replaced with woodchips.tga.

Woodchip Product Directory Structure



TRAINCAR DIRECTORY STRUCTURE

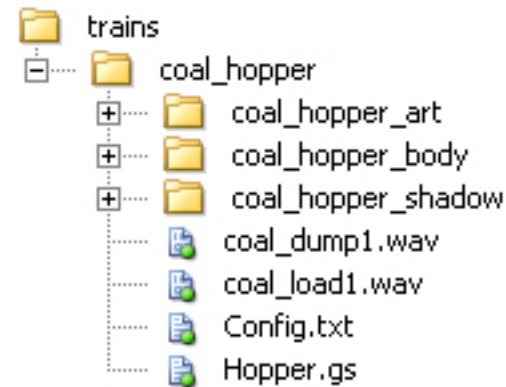
Default location:

C:\Program Files\Auran\TRS2004\World\Custom\trains

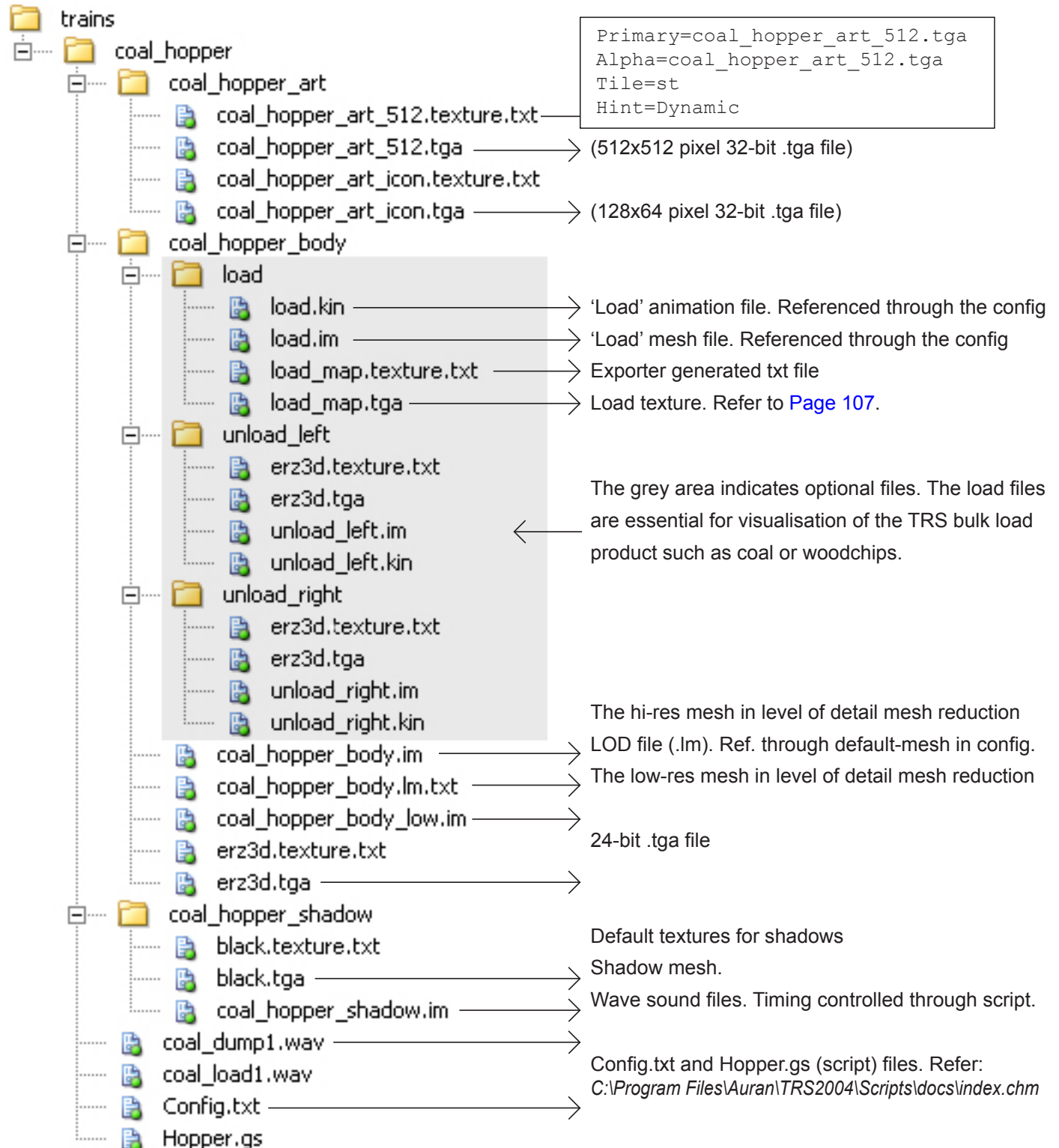
The following example is of a typical Coal hopper.

This asset has an animated load (typical for bulk load rolling stock) and animated unload doors (specific to this asset). These doors are controlled by the hopper.gs script file.

Typical Directory Structure



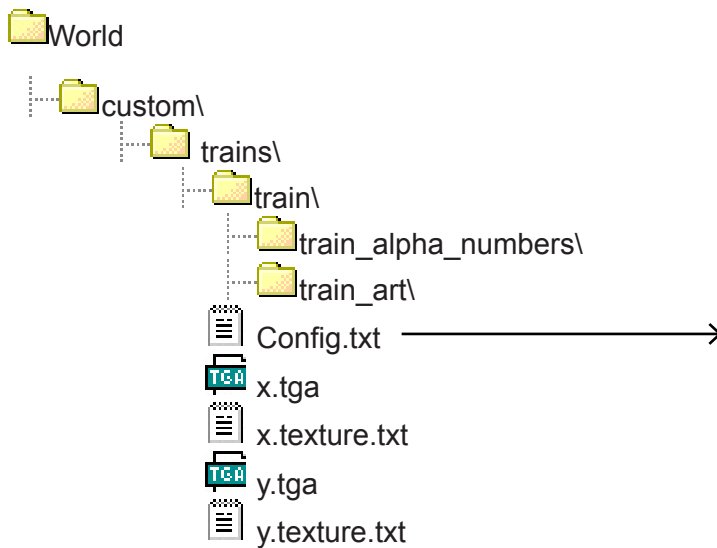
Expanded Directory Structure



ALIASING TRAINS

TRS Traincars can reference archived locomotive mesh assets for use with custom textures. This process is done by aliasing the KUID of the archived traincars.

A typical structure of an aliased loco could be as follows:



The textures must have exactly the same names and have exactly the same quantity and pixel dimensions that the aliased mesh uses.

The shadow file of the aliased loco will also be read (if present).

```
kuid <KUID2:####:####:1>
alias <KUID:-10:183>
name train
company Auran
origin AU
bogey <KUID:####:####>
engine 1
interior <KUID:####:####>
fonts 1
mass 97600
kind traincar
running-numbers
{
  rn-0 #0003
  rn-1 #0004
  rn-2 #0005
  rn-3 #0006
}
smoke_shade 0.18
smoke_random 2.5
smoke_slowlife 6
smoke_fastlife 0.8
smoke_height 1.7
smoke_fastspeed 3.2
enginespec <KUID:-1:42004209>
enginesound <KUID:-12:2100>
hornsound <KUID:-1:42003103>
description " "
kuid-table
{
  0 <KUID:####:####>
  1 <KUID:####:####>
  2 <KUID:####:####>
}
obsolete-table
{
}
username My locomotive
trainz-build 1.5
category-class AC
category-region-0 AT
category-era-0 1980s
```

The KUID of
the aliased
mesh

BOGEYS

3D STUDIO MAX AND GMAX MODEL GUIDELINES

Download Source files from the Trainz Website

POLYGON LIMITS:

Steam Loco bogey polygon recommendations
< 5000 polygons per driving wheels (including all rods and animated parts)
Diesel Loco bogey polygon recommendations
< 2000 polygons per truck.

Bogey shadow polygon recommendations
< 100 polygons per truck.

Carriage bogey polygon recommendations
< 300 polygons per truck.

Carriage bogey shadow polygon recommendations
< 100 polygons per truck.

As a general rule of thumb, less is always better! ☺

The absolute centre of bogeys should be located at World origin point (0,0,0). This is where they are inserted into the a.bog0 etc attachment points in the loco body mesh.

BOGEY TEXTURES

The materials are of Multi/Sub-Object type (one M/SO only per model) and we have used UVW Map and Unwrap UVW for texture allocation.

Diffuse Maps: Generally a single 128x128 16-bit TGA file is sufficient to texture a bogey. Additional maps (e.g. for springs) are also used.

Opacity Maps (8 bit greyscale .bmp) are supported to the same pixel dimensions as the diffuse map. Used regularly for carriage bogey sides. Reflection maps are supported but generally not used on bogey models.

Bump mapping and specular values are possible to give greater detail and variation. (3dsmax 4 + users only). See TRAINCAR Bump mapping notes [Page 100](#).

Hierarchical Sub-tree:

```
b.r.base
  b.r.wheel0
    wheel_0
  b.r.wheel1
    wheel_1
  bogey
```

b.r. helper points must be made in top view in 3dsmax/gmax.

EXPORTING MODELS:

As per 'Modeling Trains' section. Remember naming conventions and to type in the file extension under file name (e.g. TRAIN_NAME_bogey.im).

Refer to [Page 37](#) for information on reversing bogeys and animation, with reference to attachment points.

IMPORTANT NOTE: Steam Driving Bogeys

The Steam loco driving bogey is connected to the piston and physics system by adding the following tag to the bogey's config.txt: **direct-drive 1**

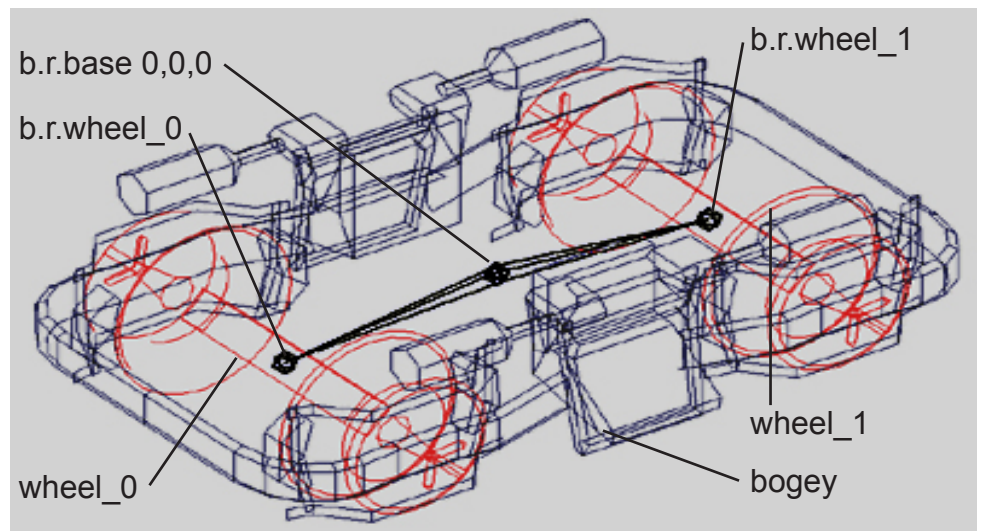
(See PB_15_bogey2 Config.txt below)

This tag MUST be included for piston and steam sounds to work.

PB_15_bogey2 Config.txt

```
kind bogey
kuid <KUID:44179:50003>
animdist 3.816
category-class AS
category-region-0 AU
category-era-0 1920s
category-era-1 1930s
category-era-2 1940s
category-era-3 1950s
category-era-4 1960s
category-era-5 1970s
category-era-6 1980s
direct-drive 1
```

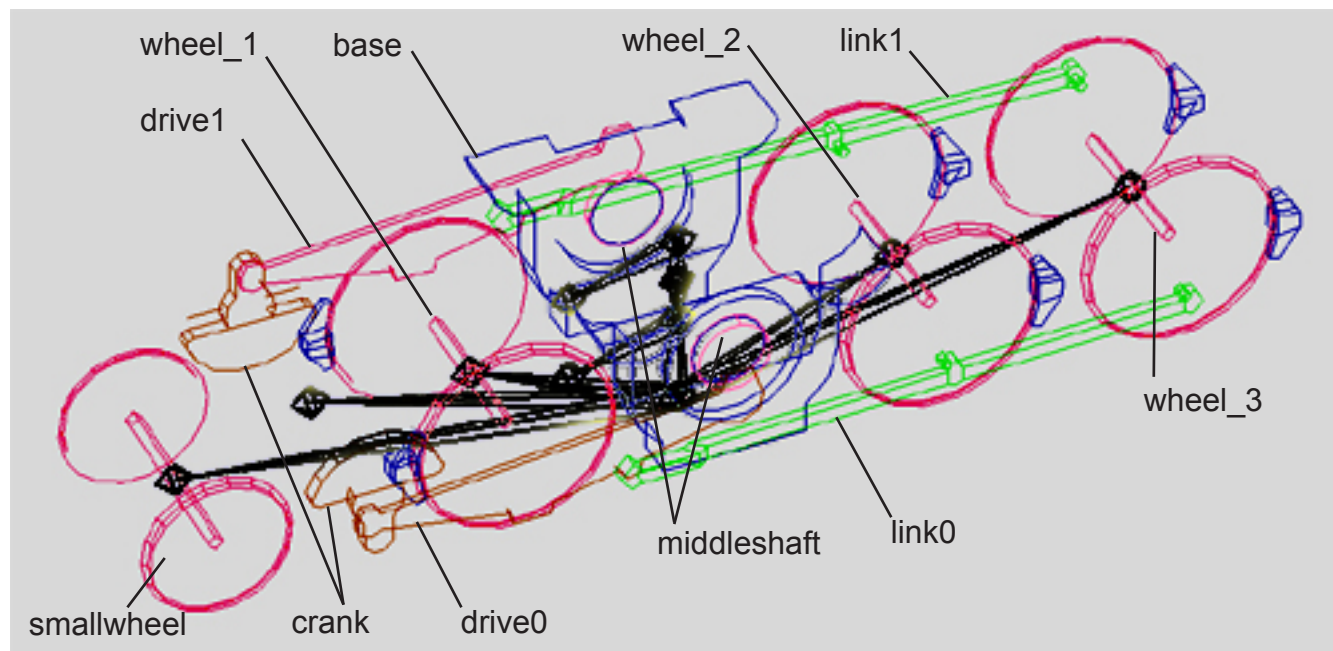
Animated Bogey Example 1



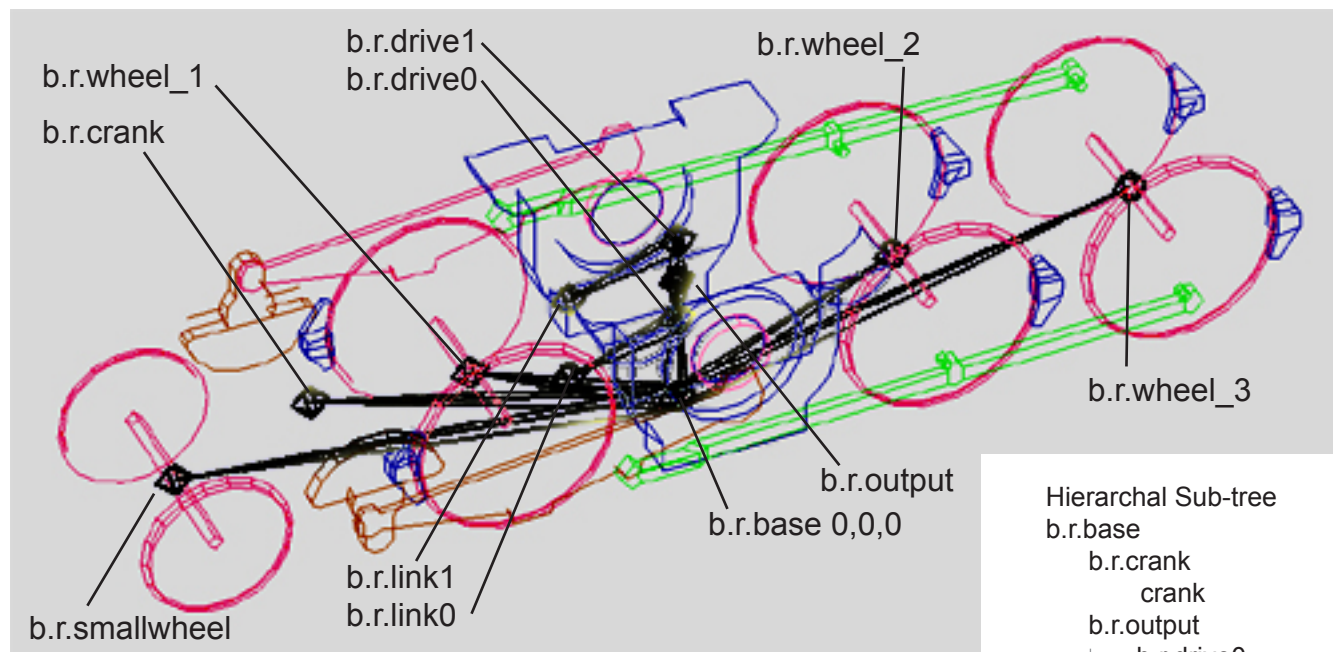
In this example, the bogey will be inserted into the Train model attachment point (e.g. a.bog0) at b.r.base (or 0,0,0). b.r.wheel0, and b.r.wheel1 (bones) were animated to turn 360° over 32 frames.

Bones must have the b.r.* naming convention for Trainz to recognise them.

Animated Bogey Example 2a - Objects



Animated Bogey Example 2b - Bones

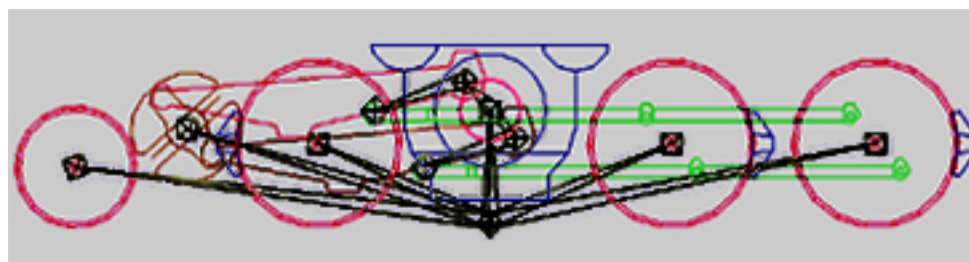


Hierarchal Sub-tree

```

b.r.base
  b.r.crank
    crank
  b.r.output
    b.r.drive0
      b.r.link0
        link0
      drive0
    b.r.drive1
      b.r.link1
        link1
      drive1
    middleshaft
  b.r.smallwheel
    smallwheel
  b.r.wheel_1
    wheel_1
  b.r.wheel_2
    wheel_2
  b.r.wheel_3
    wheel_3
  base
  
```

Animated Bogey Example 2c - Side view after a few frames



This example is much more complex than the previous example. Animation frames = 120

Wheel_1, 2 & 3 circumference = 4.2m

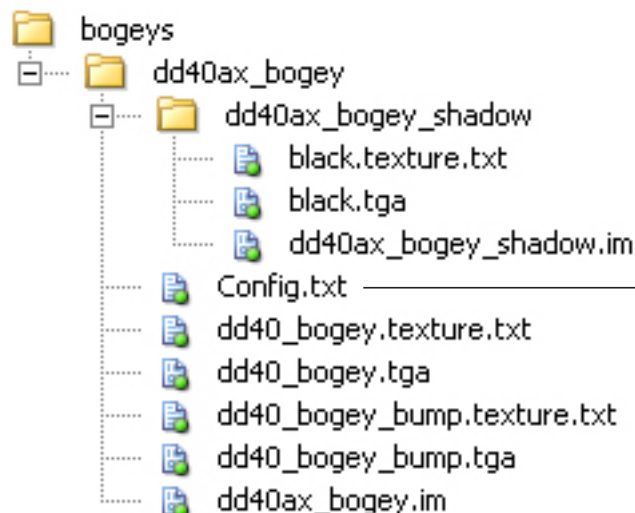
- Animated to turn 720° over 120 frames:
Distance traveled: 4.2m x 2 revs = 8.4m

smallwheel circumference = 2.8m

- Animated to turn 1080° over 120 frames:
Distance traveled: 2.8m x 3 revs = 8.4m

- Animdist: (worked out from distance travelled in 30 frames: 8.4 / 4 = 2.1) therefor **animdist 2.1**

Typical TRS bogey config.txt (.im files with mesh-table)

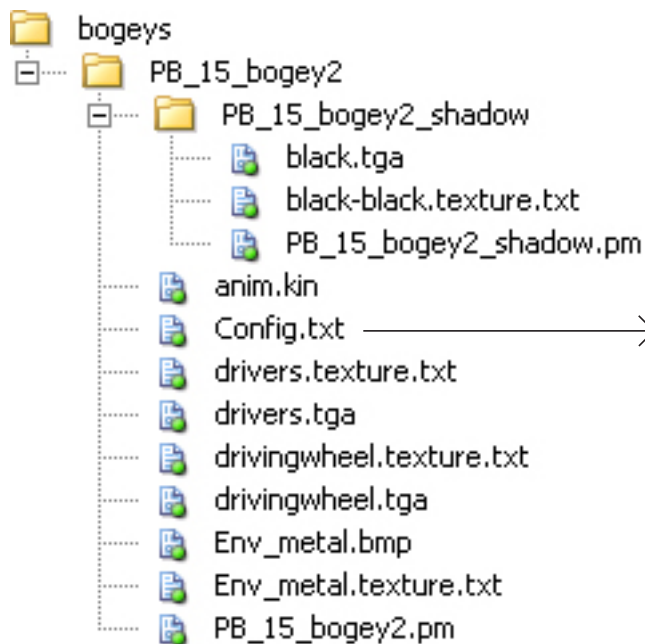


```
kuid <KUID:###:#####>
kind bogey
animdist 2.1
mesh-table
{
    default
    {
        mesh dd40ax_bogey.im
        auto-create 1
    }
    shadow
    {
        mesh dd40ax_bogey_shadow/dd40ax_bogey_shadow.im
    }
}

obsolete-table
{
}

username mybogey
description " "
trainz-build 2.0
category-class AC
category-region-0 AT
category-era-0 1980s
```

Typical UTC bogey config.txt (.pm files)



```
kind bogey
kuid <KUID:44179:50003>
animdist 3.816
category-class AS
category-region-0 AU
category-era-0 1920s
category-era-1 1930s
category-era-2 1940s
category-era-3 1950s
category-era-4 1960s
category-era-5 1970s
category-era-6 1980s
direct-drive 1
```


PANTOGRAPHS (pants)

Default location:

C:\Program Files\Auran\TRS2004\World\Custom\pants

Download Source files from the Trainz Website

Pantographs are the animated mechanisms on the roof of electric locomotives that conduct to an electric catenary (wires) above.

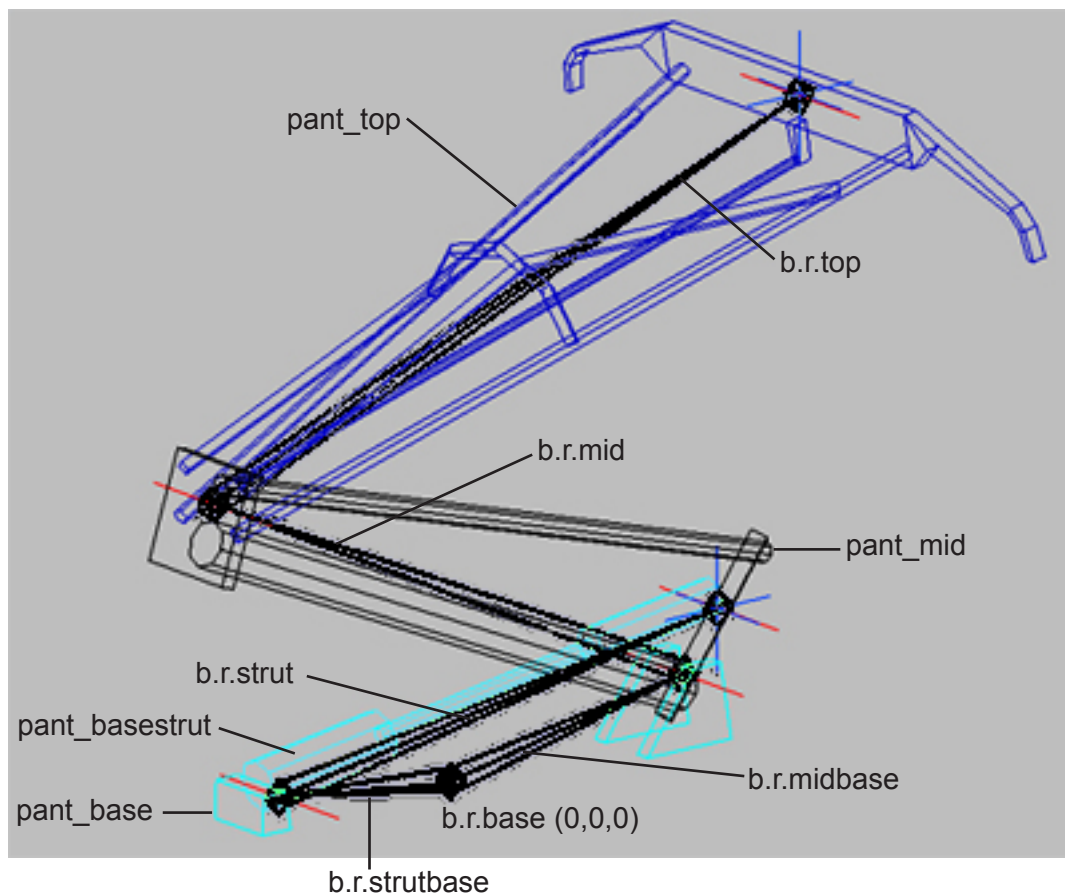
Model configuration:

Typical model configuration: (based on the bb15000 pantograph)

Generally Pantograph animations should take place over 16 frames only. Bones must have the b.r.* naming convention for Trainz to recognise them.

Refer to the *Source files* from the Trainz Website for a working example.

Note: Mesh tables do not apply to Pantographs and files are of type .pm. Animation should commence in the lowered position (frame 0) and be in the highest position at frame 16. All b.r. helper points (dummies) are constructed in top view in 3dsmax/gmax.

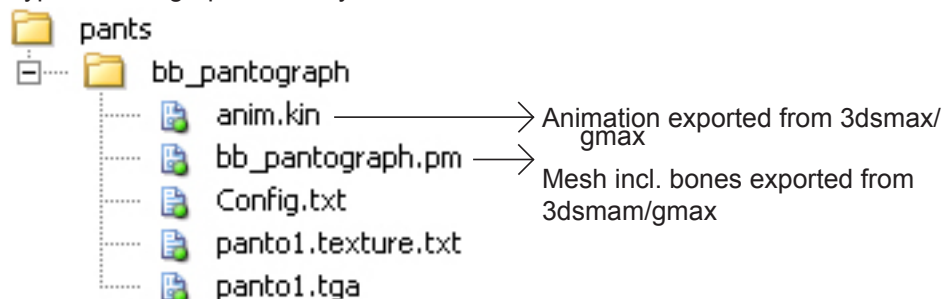


In this example, the Pantograph will be inserted into the Train model attachment point (a.pant0) at b.r.base (or 0,0,0).

Typical Hierarchal Sub-tree

```
b.r.base
  b.r.midbase
    b.r.mid
      b.r.top
        pant_top
      pant_mid
    b.r.strutbase
      b.r.strut
        pant_basestrut
      pant_base
```

Typical Pantograph Directory Structure



TURNTABLE (TRANSFER TABLE)

Download example Source files from the Trainz Website.

<http://www.auran.com/TRS2004/downloads/contentcreation/TransporterTestAsset.zip>

A transfer table is a particular kind of turntable, instead of using the in-built options of specifying angles for the track stopping positions, it uses an animation file exported from 3dsmax or gmax, with stopping points located by the key frames of the animation.

Generally the platform will move in a linear motion, but any type of motion is possible with this asset, specified by the animation file. Refer to the example Kinds in [Chapter 6](#) for full examples of turntable Kinds.

Model configuration:

A typical model configuration consists of a base model (the static pit in the ground) and a transfer table or platform model. Additional night models and other attached meshes may be used. A sample directory configuration is shown in the column to the right.

Track for the static approach tracks and the track on the moving platform are specified using the track container. In order to have the track on the platform (or any other attachment on the platform) move with the platform, a special notation in 3dsmax or gmax is required for the attachment points, as discussed on [page 137](#).

Typical Hierarchal Sub-tree in 3dsmax or gmax:

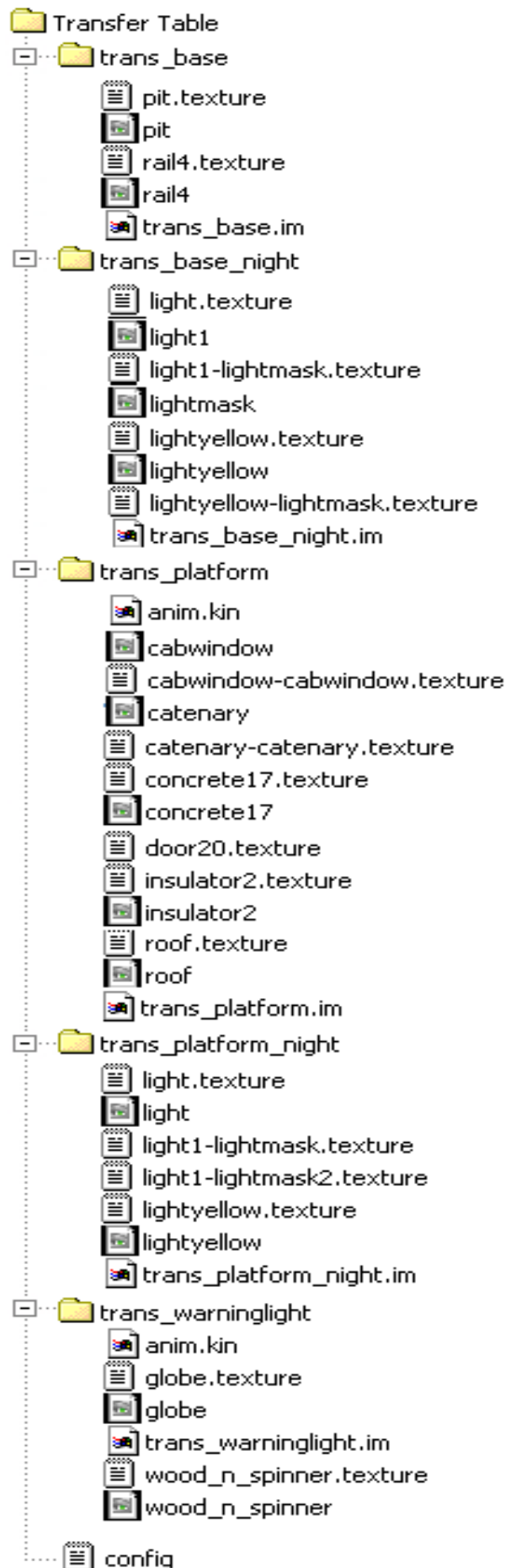
```
b.r.base
  b.r.platform
    platform
      a.r.platform/a.cabfront
      a.r.platform/a.itrack0a
      a.r.platform/a.itrack0b
      a.r.platform/a.warnlight0
  b.r.night
    night
  base
    a.otrack0a
    a.otrack0b
    a.otrack1a
    a.otrack1b
etc
```

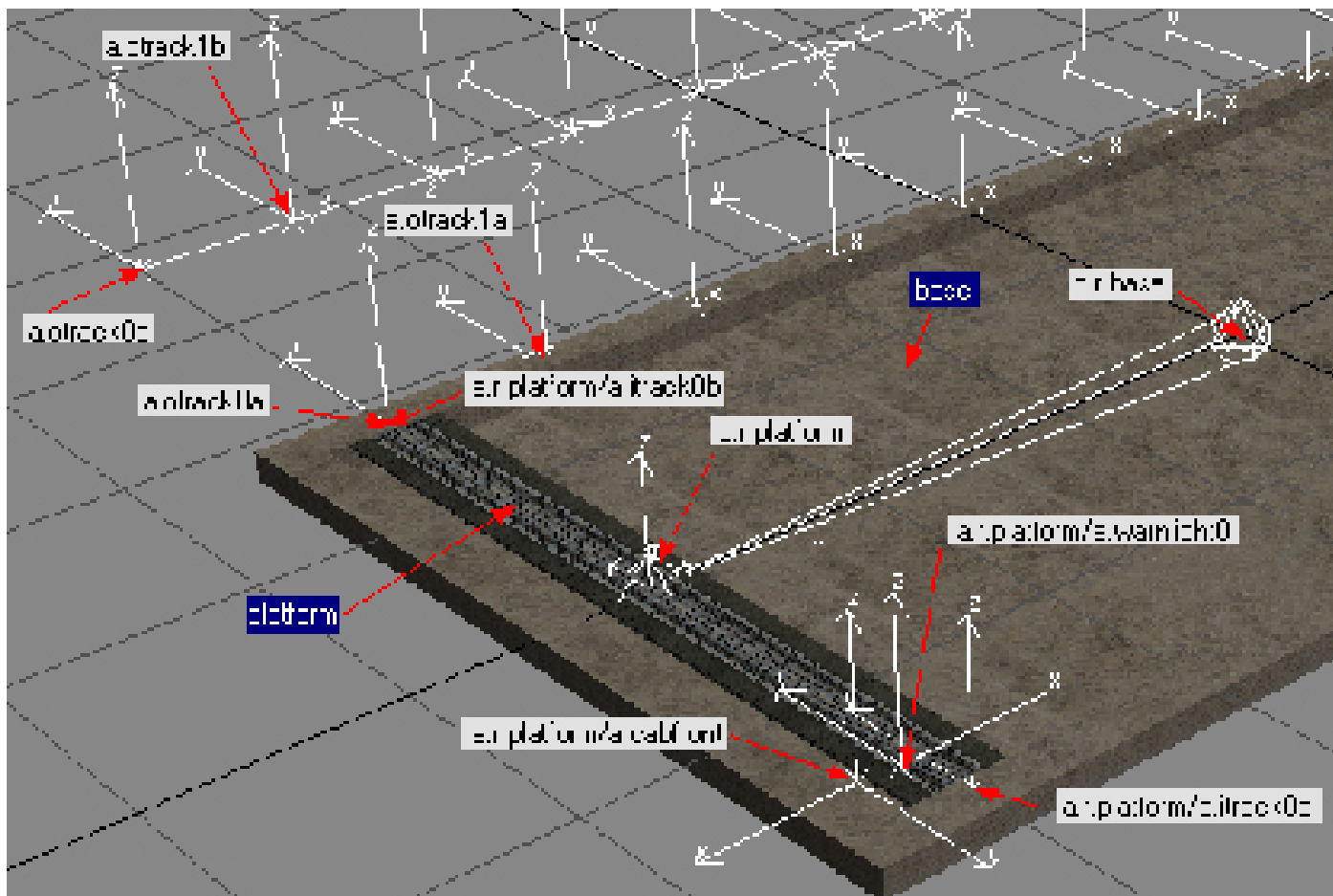
Refer to the diagram below.

In the example above, the moving platform has attached track, and interior requiring an a.cabfront attachment, and a flashing corona warning light.

The base of the transfer table, the pit, has attached track at regular intervals. A night mesh is also used for the base.

The b.r.base is the main helper point located on the origin. The moving platform is linked to the b.r.platform helper that is also linked to the b.r.base.





Note in the diagram, for the platform in the start position as shown, the attachment point on the platform, a.r.platform/a.itrack0a is in the same position as the joining track on the base, a.otrack0a.

Any object using attachment points and linked to the moving b.r.platform uses the special notation in 3dsmax or gmax shown above, the cabfront, track attachment points and flashing coronas on the platform use the a.r.platform/ notation as part of the attachment point name, however only the attachment point name itself is used in the config.txt file.

For example, the track attachment point at one end of the moving platform is: a.r.platform/a.itrack0a

The name entered in the config.txt file is: a.itrack0a

If the platform had a night mesh attached, it also would need to be attached in this manner.

Naming conventions

The naming of the track attachment points is not important in TRS2006. For circular turntables in TRS2004, the names used above were reserved names, recognised and interpreted by Trainz, to place the attached track correctly, for example, a track on the rotating turntable was a.itrack0a and a.itrack0b, the "i" denoting an inner track on the platform.

The attached track on the base was labeled as

a.otrack0a and a.otrack0b, the "o" denoting an outer track, and the "a" and "b" denoting the extremities of the track attachment.

In TRS2006, mesh tables are now used to list all the attached track, and this default notation is not now required. It has been used in this example for consistency.

Platform mesh called platform.im? for circular tables?

Animation Key Frames

An animation file exported from 3dsmax or gmax will specify the key frames that coincide with the attached track along the transfer table, for example, the key frame entries in the config.txt file might be:

```
keyframes 0,160,320,480,640
looping 0
frame-rate 30
```

The key frame 0 would co-incide with the a.otrack0a position on the diagram above, and key frame 160 would co-incide with the a.otrack1a position.

However, inaccurate stopping locations may occur in large models with many tracks. The mis-alignment may increase as you move to tracks 6 or 7 for instance until a locomotive derails. This can be corrected by entering additional key frames in 3dsmax or max, either side of the correct stopping point, for example at 158 and 162, either side of the frame 160 point. these frames are not

kisted in the key frame list in the config.txt file table. One drawback is that the platform moving from track0 to track 6 for example, will hesitate at the intermediate tracks.

Because the use of animation key frames is so versatile for a transfer table, for example a vertical mine shaft might be made, with the elevator car as the platform, the platform can vanish as it moves away from the base model.

This is caused by the platform moving outside the main bounding box of the base model. An additional tag should be used in the config.txt file, in the platform container, to correct this issue:

platform

mesh cage/cage.im
anim cage/anim.kin
use-parent-bounds 1

FIXEDTRACK

A fixedtrack in TRS could be likened to a model train sectional track system. They snap into position when moved onto another track in Surveyor.

Technically, all a fixedtrack comprises is a mesh asset with an attached track (or tracks) and surveyor only rendered arrows so the user knows where the fixedtrack starts and ends.



The model simply comprises of a few attachment points (using the *a.name* naming convention) set-up accurately in 3dsmax or gmax, and a single invisible polygon to allow exporting, and for in-game asset selection.

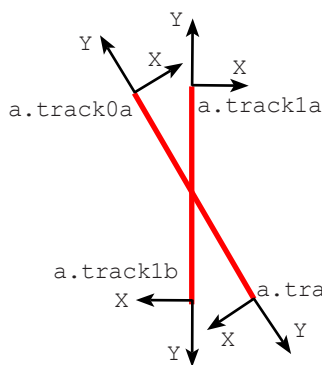
Note that correct track end attachment orientation is essential. The Y axis must point 'out' at the correct angle. The Z axis must point 'up'. Mid points only need to be in the correct spline path. See diagram below.

When a spline track is attached to a fixedtrack the fixedtrack will update to the attached track type. (unless *useadjoiningtracktype 0* is used as shown left)

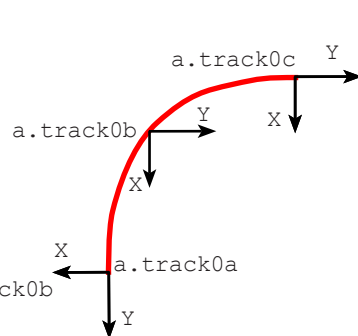
The arrows are inserted at each end as a *kind attachment* - referenced by the arrow's KUID: <KUID:-3:10092>

Each fixedtrack asset needs a preview-mesh, as spline tracks will not render in the Preview window. A preview-mesh can simply be setup as a *kind mesh*. This way the preview-mesh will never be selectable or seen in Surveyor.

Crossing Attachments



Curve Attachments



TRS2004 released fixedtracks comprise of only curved and straight sections. Crossings may be made, just create two attached-track fields.

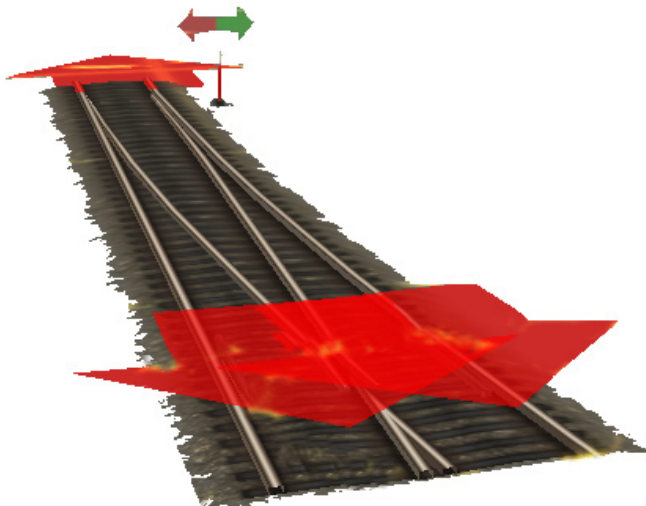
FIXEDTRACK - Junctions

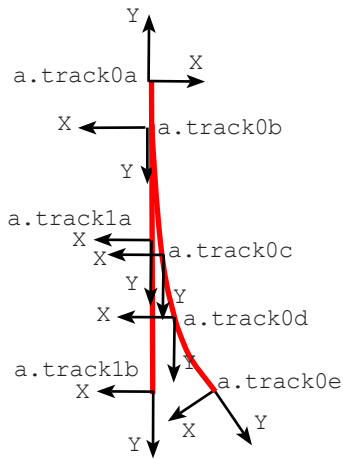
Junctions (turnouts) are now possible in TRS2006, with the use of additional tags. A number of samples are available in TRS2006. because the Kind fixed track is not based on the Kind Track asset, there is some functionality for the common flexi track that is not available for the fixed track asset.

The default red and green direction arrows of the flexi track junctions are not available with the fixed track object. The fixed track object may be raised or lowered and placed on a slope. Operation of the turnout using the mouse will only operate one trigger or animation. Multiple operations such as double slip junctions should be possible by scripting.

The example asset in TRS2006 has been constructed with a main default mesh consisting of the fixed rails and ballast, the moveable blades are a separate animated mesh connected to an attachment point in the main mesh, and a separate lever mesh also connect to an appropriate attachment point either to the left or right of the default mesh.

In the example, the red and green arrows have been simulated using an arrow texture with some transparency, with rotating animation, and attached to the lever attachment points. By amending the config.txt file, the arrows and the levers may be deleted or replaced by other suitable meshes.





a.track0b is the common track attachment point. The curved track may require more attachment points than the straight track, to define the curve shape.

Note the Y axis at the end attachments must point outwards and be aligned with the track entrance or exit direction.

To define the attached track in attached-track container, attachments a.track0a and a.track0b are common to both track tag lists.

The vertices are listed as:

for the curved track0

- 0 a.track0a
- 1 a.track0b
- 2. a.track0c
- 3 a.track0d
- 4 a.track0e

and for the straight track1

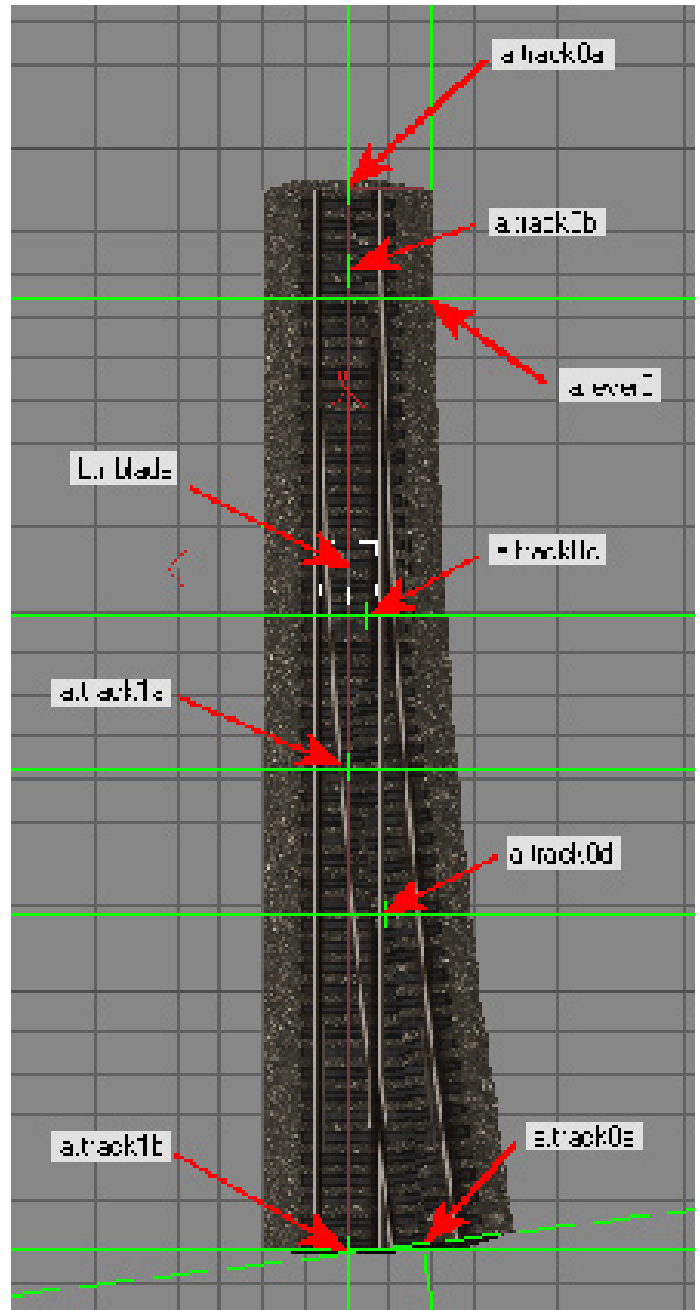
- 0 a.track0b
- 1 a.track1a
- 2 a.track1b

The diagram in the column to the right shows the positioning of:

the various track attachments (referenced in the attached-track container);
the helper point b.r.blade for the animated blades; and
the switch lever attachment, a.lever0, called the junction-vertex tag in CCP, and the lever mesh, referenced by the junction-lever-mesh tag in CCP.

For convenience, the b.r.blade is placed on the origin, and is also the helper for the default mesh (the fixed parts of the turnout).

For examples of the attributes and tags required for Fixed Track assets, refer to Kind Fxed Track Containers, Tags and Examples in [Chapter 6](#) and [Chapter 7](#).



CHUNKY MESH TRACK

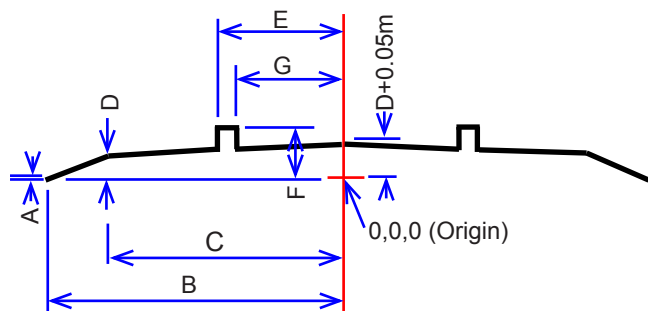
Chunky mesh track is an asset that uses a default inbuild mesh. The creator does not have to construct a mesh in 3dsmax or gmax.

The cross sectional track and ballast shape of the mesh is pre-defined by the diagram on the next page.

A 128 by 128 pixel tga file is used for the texture. This generally includes the ballast, ties and a larger rail section texture on the right of the graphic. Because the shape is defined by geometric proportions, the approximate equivalent dimensions measured in pixels is shown in the chunky_info texture file data shown below. This is approximate due to rounding of values to whole pixels. For additional information on using the Kind, reffer to [Page](#)

chunky_info

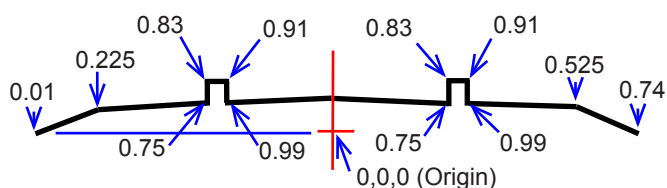
These values (in metres) define the shape of the mesh created for the track. See drawing below:



chunky_info A, B, C, D, E, F, G
 chunky_info 0, 2, 1.2, 0.2, 0.85, 0.3, 0.7

chunky_info texture file

The texture file with the track texture on the left and a rail texture on the right is 128 x 128 uncompressed tga, and may have an alpha layer. The texture is mapped to the mesh shape above using the values in the drawing below, as fractions of the 128 pixel width.



Fraction	0.01	0.225	0.525	0.74	0.75	0.83	0.91	0.99
Pixels	1	29	67	95	96	106	117	127

Splines

Splines have a number of tags that have special effects in Trainz, and also need to be constructed in a certain way in 3dsmax or gmax.

For example, the effect of the following tags are explained in the diagrams:

upright 0

This effects how vertical the objects in the spline are, for example a row of poles:

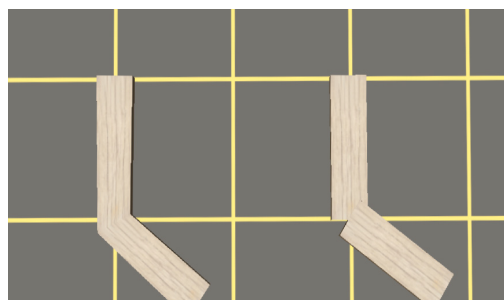
0 = the poles will be placed at right angles to the slope of the ground.

1 = the poles will be truly vertical regardless of the ground slope.

bendy

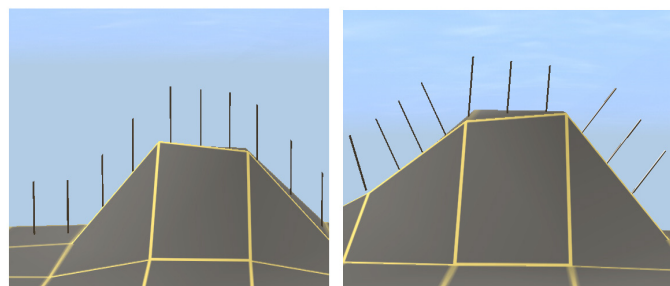
Switches how track is bent on corners, set as 1 allows the mesh to be deformed as the spline is bent around corners.

Notes: bendy and upright have a visible effect for Kind Track splines, see diagram below. For Kind Bridge or Tunnel, the splines show as in bendy 1, bendy 0 has no visual effect. However, bendy 1 should always be entered in the config.txt file for bridge and tunnel Kinds, as the tag improves handling of the spline and Trainz performance .



Bendy 1

Bendy 0



Upright 1

Upright 0

length and endlength

A spline may be made by specifying a length value in the config.txt file. To add a different model at the start or end of the spline, Initiators and Terminators are used.

Initiator

Name of model to use at start of bridge, placed in sub folder with same name.

terminator

Name of model to use at the end of the bridge, placed in a sub folder with same name.

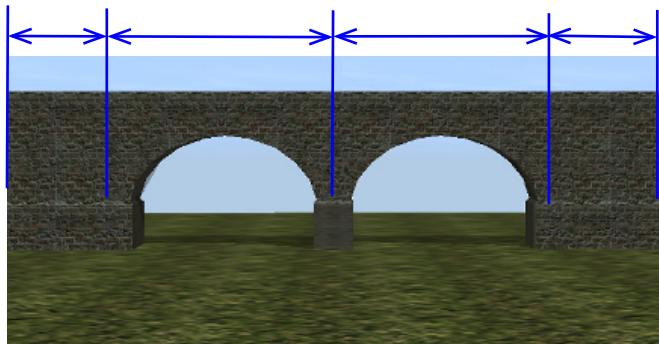
length

Length of the main spline track segment in meters.

endlength

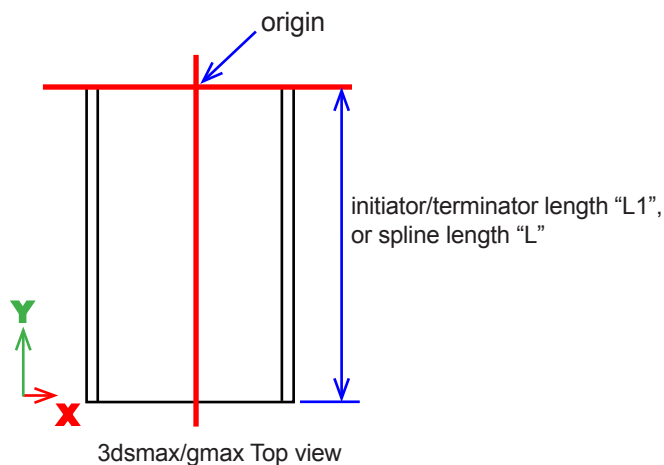
Length in meters of the initiator and terminator models.

initiator "L1" spline "L" spline "L" terminator "L1"



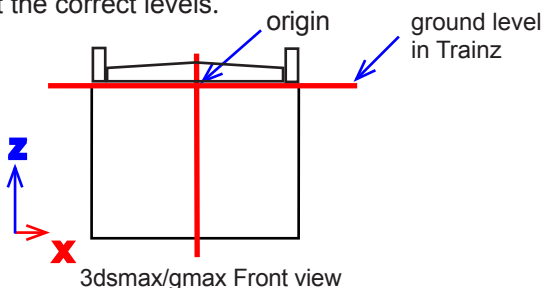
The initiator or terminator is a fixed length and the main spline model repeats as in the diagram above.

In 3dsmax/gmax, the initiator, terminator and spline models must be constructed starting on the origin and extending in the negative Y axis direction. The top view in the diagram shows the correct placement and dimensions L1 or L.



The initiator may be rotated 180 degrees to create a terminator model, if required.

Attachment points will be automatically generated in Trainz at ground level. The model heights need to be adjusted in 3dsmax/gmax so a road or track will connect at the correct levels.



For additional notes on splines refer to [Page 84](#).

CHAPTER 9

Uploading to the Download Station

The purpose of this chapter is to familiarise users with the new Uploading and Content Distribution procedures made available by Content Creator Plus.

The Trainz Download Station

With the introduction of Content Manager Plus (CMP), many aspects of the upload process have been changed somewhat.

Unlike previous versions of Trainz, uploading and downloading are now fully encapsulated within the Content Manager Plus (CMP) program. Assets no longer need to be manually packaged in order to be uploaded, making the process much simpler.

To upload your content, you must have a valid Planet Auran profile. You are also only able to upload your own custom content.

Steps to Upload

Verify Content is Error Free

Before uploading content it must be verified and found error-free; a process which is rigorously enforced by the CMP program. This verification occurs automatically during the upload process in order to ensure a high standard of quality among download station assets.

Faulty assets are marked with a  in CMP. If an asset is faulty, it must be fixed with the Content Creator Plus (CCP) program and verified as error free before it is able to be uploaded.

Select Content to be Uploaded

Next, you must select the assets to be uploaded. You needn't select the dependencies of an asset you wish to upload as these will be selected automatically by the program.

If some of your dependant content has been created by other authors, or if it is already present on the Download Station it will automatically be removed from your upload list.

When a user downloads your content, any depend assets which are present on the Download Station will also be queued for download.

Begin Uploading

Once you've selected the appropriate assets to be uploaded, clicking the "upload" button will bring up the Planet Auran License Agreement which details the legal issues regarding uploaded content.

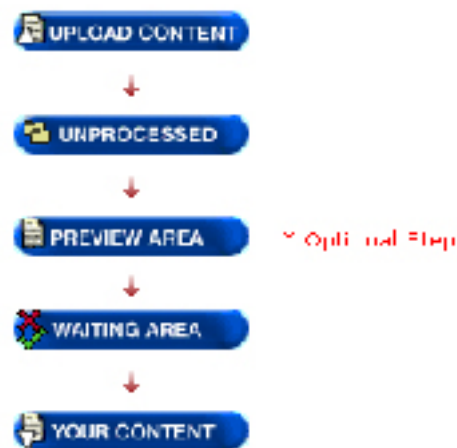
Accepting this agreement will upload your content to the Download Station and open the Planet Auran "Your Content" page.

This page is part of your personal Planet Auran profile and allows you to manage your online content at any time.

You can access this section separately by visiting the Auran website and following the "Planet Auran" link.

The "Your Content" section is divided into the following

areas:



Upload Content: You needn't utilize this section as CMP now handles the entire upload process. It is recommended that you only use CMP to upload TRS2006 assets.

Unprocessed: The "unprocessed" section shows the files you have uploaded which have not yet passed through the automated checking process.

After uploading an asset via CMP you will find it in this section. Any files you upload using CMP will be packed into an "upload.cdp" file automatically. If you upload many files in one instance, they will be compiled into a single file before they are processed online.

Files are usually processed daily and will remain in this section until they are either verified as valid or rejected. If you have uploaded a file and decide that you no longer wish for it to be processed, you can mark the offending upload as "do not process". The file will remain in this area should you change your mind, and will be purged at the end of the day.

Preview Area: The preview area acts as a temporary 'holding bay' for previewing and testing your content before it is submitted for approval. The main features of the Preview Area are as follows:

- You can review and approve your own content before the final approval process.
- You can send a link to your content to others for testing purposes. Only you have initial access as the content is not publicly available until you submit the content for approval.
- You can re-check all the Download Station pack information before final approval, i.e. Pack contents, category, obsoletes, descriptions etc.
- You may have multiple revision uploads for assets of the same KUID.
- You may delete the content before it is made publicly available.

Important note:

Preview Area content can only be held for 4 weeks since last revision. All testing/revisions must be done within this time.

If you have (say) 4 assets for upload - A locomotive, an engine file, a bogey and a pantograph, and you only update ONE of these assets, the other 3 are still on the original 4 week period.

Waiting Area: The waiting area section houses that content which has successfully passed through the automated checks made by the Download Station system and is awaiting visual confirmation from internal Auran staff. This is done for a number of reasons, including:

- Verifying that the name and description are valid and appropriate
- Verifying that the correct category is selected
- Verifying that the thumbnail image is visually acceptable

You will receive an email to indicate if your content has been approved or declined. If approved, the content will be available on the Download Station approximately 8 hours after the email is sent.

Your Content: The “Your Content” section shows your content which is currently available for download on the Trainz Download Station.

Download Station Checks

If your content fails any of the following checks, the content will be removed and you notified:

- File/s were extracted successfully.
- Compares the number of files to the number processed
- Ensures that the User ID of the content belongs to the user uploading the content
- If content was uploaded via a Group member, it checks that the User ID belongs to that Group.
- That the KUID is valid
- Name for minimum length of 3 characters and no swear words.
- Description is present and free of swear words. If updating content, it checks that the content being updated is the latest version and not a previous version.
- The Region / Country codes are valid.
- The Eras are valid.
- Description for maximum length of 2048 characters, if over 2048, description is truncated.
- Name for maximum length of 64 characters, if over 64 the name is truncated

- Valid thumbnail tags and accompanying image are included

- File size is below the maximum. The maximum file size will be stated on the Download Station upload page as this may be subject to revision.

Packaging Files (CDP's)

You may wish to distribute your content through means other than the Trainz Download Station. To facilitate this, CMP allows you to export CDP “package” files (which experienced users may already be familiar with).

Like the upload process, when packaging CDPs you can only package your own custom content for distribution.

- To create a package with CMP:

- Select the items of content you wish to add.

- Right Click and select “Save to CDP” (or press CTRL SHIFT D)

Save the resulting CDP file to a location of your choice.

The CDP format is intended for content distribution purposes and doesn't automatically include any dependencies associated to the assets you are packing.

If you wish to store backups of your work then it is recommended that you use the “Archive” CMP feature instead (which is mentioned in the CMP help file).

CHAPTER 11

Particle Effects and Scripts

INTRODUCTION

TRS gives you the ability to add customizable smoke, steam, vapor and similar effects to your custom trains and scenery objects. For simplicity, this document will refer to this set of effects as simply smoke effects.

There are two ways of setting up particle effects (pfx) for TRS mesh assets.

1) As per UTC. Setup all the settings and variables through the config.txt. Not recommended as future versions may use twinkles exclusively.

or the recommended/preferred method...

2) Use a new pfx tool called Twinkles PFX to set pfx parameters and reference it via the config.txt.

It is assumed the reader is already familiar with creating and exporting models from either 3dsmax or gmax.

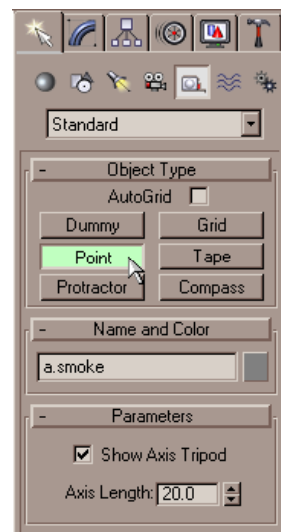
Method

Smoke effects are added to custom trains and scenery objects in two steps:

1. Add attachment points to the original model.
2. Add smoke tags to the object's config.txt file.

Adding Attachment Points

Attachment points are added to the original model using 3dsmax or gmax wherever a smoke effect is desired. See figures 1 and 2 below to locate the Insert Point tool. After a point is inserted, it must be given a name with a prefix of 'a.' to identify it as an attachment point, e.g. a.smoke, a.steam, a.safety, a.mist, etc.



3dsmax insert point



gmax insert point

The attachment point should also be rotated so that its Y axis is pointing in the direction that smoke particles will be emitted. (Ensure Axis Tripod is checked to see the point's orientation - use the Hierarchy - Affect Pivot Only option in 3dsmax or gmax.) When finished, save and export the model as per normal.

Make sure you rotate the point itself and not the axis when aligning the point. Unselect the Hierarchy - Affect Pivot Only option before rotating.

PFX FROM CONFIG.TXT

(NOT USING TWINKLES - NOT RECOMMENDED)

ADDING SMOKE TAGS

Smoke blocks are added to an object's config.txt file to describe each smoke effect that will be created on the object. Smoke blocks are named smoke# (where # is a number) and are sequentially numbered starting at 0. See Example 2.

Smoke blocks have two sections: main and sequence properties. Main properties describe the attributes that do not change based on the mode's key. Sequence properties describe a set of one or more phases/periods in the smoke emission sequence.

A smoke block has the following format:

```
smoke#
{
  mode           time | speed | anim | timeofday
  attachment     <name of attachment point>
  color          <red>, <green>, <blue>, <opacity>
  accel          <x>, <y>, <z>
  loop           <n>
  start          <n> [, <n>] ...
  period <n> [, <n>] ...
  rate           <n> [, <n>] ...
  velocity       <n> [, <n>] ...
  lifetime       <n> [, <n>] ...
  minsize        <n> [, <n>] ...
  maxsize        <n> [, <n>] ...
}
```

Notation:

- # Is a number, starting with 0
- [] Means optional,
- ... Indicates a variable number of parameters,
- | Means or.

Breakdown:

<name of attachment point>

Is the name of an attachment point in the model. e.g. a.smoke, a.steam, a.chimney etc

<red>, <green>, <blue>

Are numbers from 0 to 255 describing the intensity of that color component.

<opacity>

Is a number from 0 to 255 describing the effect's initial opacity / transparency.

<x>, <y>, <z>

Are vector components pointing in the direction of the sum of all forces affecting this smoke effect. Essentially, <z> describes gravity, and <x>, <y> describe the force of wind.

<n>

Is a decimal number.

MAIN PROPERTIES:

mode

Describes the mode or type of this smoke effect. This affects how start and period are interpreted. Default is time. In all modes, period can be set to -1 (default) to imply the phase is active until the next phase begins.

If set to **time**, start is a set of time values in seconds after the creation of this effect's parent object when this phase of the effect will start. Period is the duration of time this effect will remain active. Scenery objects currently only support time mode.

If set to **speed**, start is a speed in meters per second (m/s) and period is not used. (Note: 1 m/s = 3.6 km/hr.) All other sequence attributes (rate, velocity, lifetime, minsize, maxsize) are interpolated so there are smooth transitions between phases. See smoke3 in Example 2.

If set to **anim**, start is a value from 0.0 to 1.0 which describes the start time into the object's animation cycle. period is a value from 0.0 to 1.0 that describes the duration over which the effect is active. start + period must not exceed 1.0.

If set to **timeofday**, start is a value from 0.0 to 1.0 which describes the time of day when this effect will start.

Values range as follow:

0 - midnight, 0.25 - 6am, 0.5 - midday, 0.75 6pm, 1.0 - midnight.

color

The color of the smoke effect. e.g. '150,150,150,255' for dark smoke; '255,255, 255,150' for steam; '150,150,255,255' for water. Default is '255,255,255,255'.

accel

Acceleration. A vector pointing in the direction of the sum of all forces affecting this smoke effect. Essentially, <z> describes gravity, and <x>, <y> describe the force of wind. Default is 0,0,0.

loop

Time in seconds to loop the smoke sequence. Only valid if mode is set to time. Not functional in TRS.

SEQUENCE PROPERTIES:

The following properties can be set to a single value or a set of values for multiple phases of the smoke effect. Please note that phases must not overlap as only one phase can be active at any one time. If a property has a set of values, it must be the same length as **start**. If a single value is given then it will be used for all phases of the effect. See Example 1 using multiple phases.

start, period

See mode.

rate

The rate of emission in particles per second for modes time, speed, and timeofday, or the number of particles to emit over the animation period for anim mode. Default is 4.

velocity

The initial speed of emitted smoke particles. Default is 1.

lifetime

Time in seconds that smoke particles exist for. Default is 3.

minsize

Start size of smoke particles. Default is 0.

maxsize

End size of smoke particles. Default is 3.

In general, it is better to use a low emission rate with large particles (ie min/max size) than using a high emission rate with small particles to reduce the impact on frame rate. Smoke effects can be quite stunning but are best used in moderation.

Try experimenting with the different values to get a feel of how they affect the smoke effects. Many different types of effects other than smoke are possible with only a little imagination, e.g. waterfalls, mist, toxic green clouds, fire by using a few effects at the same position to simulate the smoke and flames etc.

EXAMPLE 1 - SMOKE FROM A FACTORY'S CHIMNEY

Using a model of a factory with a chimney, an attachment point called 'a.smoke' is placed at the top of the chimney with it's Y axis pointing up. The factory is then exported as an indexed mesh (.im file type) to the Trainz\world\custom\scenery\factory folder and the model's art assets are copied to the same location. The following config.txt file will cause smoke to come out of the factory's chimney between 6am and midday and 3pm and 6pm. Please note the given KUID is invalid and should not be used in your own custom context.

Config.txt

```
kuid <KUID2:####:####:1>
region Britain
kind scenery
type Industrial
light 1

smoke0
{
    attachment a.smoke
    mode timeofday
    color 150,150,150,250
    accel 1,0.3,0
    start 0.25, 0.5
    period 0.25, 0.125
    rate 8
    velocity 3
    lifetime 5
    minsize 0.5
    maxsize 2
}
```


EXAMPLE 2 - STEAM TRAIN

An animated steam train model that requires four smoke points may be set up as follow:

- Dark smoke from the main chimney stack that is dependant on the trains velocity (a.smoke, Y pointing up),
- A constant steam trail from a small safety pipe on top (a.steam.safety, Y pointing up),
- 2 steam trails on each side of the train that alternately expel steam keyed to the animation of the trains wheels (a.steam.l, a.steam.r, Y pointing outwards).

The model is exported as an indexed mesh (.im file type) to 'Trainz\world\custom\trains\steam_train\steam_train_body' folder and the model's art assets are copied to the same location. Please see the custom content creation guide for more information on creating your own custom trains. The following config.txt file in the parent folder will generate the desired smoke effects. Note the given KUID is also invalid and should not be used in your own context.

For example purposes, the settings of an F7 train have been used.

TWINKLES PFX

(RECOMMENDED)

Twinkles pfx is a particle effects editor.

Twinkles was designed to simplify the creation and configuration of 3D effects for use in 3D games. Twinkles is game independent and requires a TRS effects plugin to convert it's particle emitters into TRS format.

Once installed, adding an effect in Twinkles allows you to add and configure the TRS particle emitter effect.

Full documentation is supplied with Twinkles PFX as is the TrainzPFX plugin.

Examples of smoke block configuration for TRS assets are available from the Twinkles PFX User Guide.doc which forms part of the Twinkles install.

Download Twinkles from the following site:

<http://www.auran.com/TRS2004/downloads/contentcreation/Twinkles.zip>

Config.txt

```
kuid <KUID2:####:#####:1>

kuid <KUID2:####:#####:1>
kind traincar
bogey 0
engine 1
name Steam Train
mass 100000

enginespec <KUID:###:#####>
enginesound <KUID:###:#####>
hornsound <KUID:###:#####>
interior <KUID:###:#####>

smoke0
{
    attachment a.steam.l
    mode anim
    color 255,255,255,150
    start 0
    period 0.4
    rate 2
    velocity 1
    lifetime 2
    minsize 0.05
    maxsize 1
}

smoke1
{
    attachment a.steam.r
    mode anim
    color 255,255,255,150
    start 0.5
    period 0.4
    rate 2
    velocity 1
    lifetime 2
    minsize 0.05
    maxsize 1
}

smoke2
{
    attachment a.steam.safety
    mode time
    color 255,255,255,150
    rate 2
    velocity 1
    lifetime 2
    minsize 0.05
    maxsize 1
}

smoke3
{
    attachment a.smoke0
    mode speed
    color 100,100,100,200
    start 0,10,20,30
    rate 3,5,7,9
    velocity 3,4,5,5
    lifetime 4,3,2.5,2
    minsize 0.3
    maxsize 2
}
```


SOUND SCRIPTS

Soundscripts give ambient or directional sounds to objects. They cannot be used on track, bridge or spline objects. Wav files should be located within the same directory as the config.txt file. Examples as follows:

MOJUNCTION

Config.txt

```
kuid <KUID:###:#####>
kind mojunction
region Australia
trackside 2
light 1
mesh-table
{
  mode0
  {
    mesh lever1/lever1.im
    auto-create 1
  }
  mode1
  {
    mesh lever2/lever2.im
  }
}
soundscript
{
  toggle
  {
    trigger toggle
    distance 5, 100
    nostartdelay 1
    repeat-delay 1
    sound
    {
      points.wav
    }
  }
}
etc.
```

PEOPLE CROWD

Config.txt :

```
kind scenery
region Australia
kuid <KUID:###:#####>
type People

soundscript
{
  daysingle
  {
    repeat-delay 0
    distance 3,150
    sound
    {
      crowd_1.wav
    }
  }
}
etc.
```

MAP

Config.txt

```
kind map
kuid <KUID:###:#####>
soundscript
{
  morning
  {
    ambient 1
    value-range 1, 0.1
    volume 0.3
    sound
    {
      ctry_day_1.wav
    }
  }
  night
  {
    ambient 1
    value-range 0, 0.9
    volume 0.3
    sound
    {
      night_loop.wav
    }
  }
}
username Britain
workingscale 0
workingunits 0
water <KUID:-1:8009>
region Britain
etc.
```

THUNDERBOX

Config.txt

```
kuid <KUID:###:#####>
region Australia
light 1
kind scenery
type Residential

soundscript
{
  dayloop
  {
    repeat-delay 15,50
    distance 5, 50
    sound
    {
      strain_1.wav
    }
  }
}
etc.
```

Breakdown of Soundscripts:

repeat-delay

1 or 2 numbers (min, max, in sec) time to delay between the end of the sound playing, and playing it again randomised between (min .. max)

default min = 0 default max = min

attachment

Attachment point on the object to which the sound is attached.

Default: The sound will attach to the origin of parent object (not used for ambient sound)

distance

2 numbers (meters)

1st number = the distance at which the sound is played at 100%

2nd number = The cut-off distance. Doesn't affect the volume of the sound default: 50m, 150m

sound

List of .wav files to play (randomly picked)

volume

*Gain of the sound
Default 1.0 = 100%*

ambient

0 or 1, default 0 is off

Ambient sounds have no 3D "position" and may be stereo. Non-ambient (positional) sounds are positioned on the object and must be mono - see attachment above

value-range

2 numbers, currently used only for day/night sound effects.

Midnight is 0.5, midday = 0.0 or 1.0

Where the numbers are not the same, this sets the start and end times for the sound to play.

Default 0,0 (off)

trigger

A trigger may be used in an event file (.evt) associated with an animation. It plays at selected key frames of the animation as defined in the event file. Sound files may be triggered in this manner, and from scripting.

Used in the mojunction example (switch lever) the "toggle" action is automatically triggered when the lever is operated and the sound plays. The sound doesn't play until the trigger message happens, as a result of the lever operating.

nostartdelay

0 or 1, default 0

If not set, the sound will have a short delay before playing, this stops flanging (flanging is a really nasty sound caused when several copies of the same sound are played at once).

dayloop, daysingle, morning, night

These have no current function in Trainz and have only been put in for user reference.

Each is a single word only. Do not use a space.

TRAINZSCRIPT TUTORIAL

Introduction

Welcome to the first TrainzScript tutorial.

TrainzScript is the scripting language developed for Auran Trainz. This document will teach you how to create a very simple scenario - it does not aim to teach you how to program, or teach you programming concepts. TrainzScript may be used from Version 1 Service Pack 3 onward to create scenario content.

If you do not understand programming concepts, you may need to read further tutorials before trying to create Scenarios for TRS. We will be releasing a user-friendly interface for compiling powerful scripts in a later version.

Please take the time to follow the steps of this document from start to finish. The tutorial is presented in an informal (non text book) manner. More help can be found by visiting the Scenarios forum at <http://www.auran.com/trainz/forum/default.htm>

Where do I find TrainzScript?

TrainzScript is a scripting language used to drive the scenarios. Each scenario will have one or more TrainzScript files (.gs) located in its directory in the **World\Custom\Scenarios** folder. The supporting TrainzScript files are found in the **\Scripts** folder. The script files in this folder give you all of the supported functions required to control Trainz. Over time, you will learn most of the functions provided in these files. The TrainzScript compiler (gsc.exe) is located in the **\Bin** folder.

Go to the bin folder in your dos prompt, and run the compiler with the -d flag as follows.

```
gsc -d > reference.txt
```

This will copy the TrainzScript documentation to the file reference.txt. Consult this document as a reference manual for the TrainzScript language.

Creating A Scenario

Lets create our very first scenario. This will teach you how to use the TrainzScript compiler and guide you through the process required to make a simple scenario that just loads a map.

1. Launch Trainz, and create a new map in Surveyor. Lay a small track loop, and place a track mark named "START" somewhere on your track. Make sure you name the track mark using UPPERCASE, as all named object in TrainzScript are case-sensitive. Place a lever somewhere on the track so we have something for the camera to focus on. Save your map and call it "Tutorial1".

2. Before exiting out of Surveyor, select the "Export Scenario TSO" from the Trainz Main Menu, and type in "Tutorial1".
3. Quit out of Trainz go to your Trainz\World folder. Locate the config.txt file in the folder World\Custom\Maps\Tutorial1.
4. Open the config.txt file of your Tutorial1map. It should look something like the following.

```
kuid <KUID:-2:2023211879>
kind map
username Tutorial1
workingscale 0
workingunits 0
water <KUID:-1:110015>
region Australia
```

Note the KUID for your new layout. In this case, it is -2:2023211879.

5. Now go to the World\Custom\Scenarios\Tutorial1\ folder and open the config.txt file of your Tutorial1 Scenario. It should look something like the following:

```
kind activity
username Tutorial1
scriptlibrary Tutorial1
scriptclass MyTutorial1
kuid <KUID:-2:2023211880>
kuid-table
{
tutorial1 <KUID:-2:2023211879>
}
description "Tutorial1"
```

A few notes on this. The KUID-table is a name-KUID translation table used by the scenario. All objects loaded by the scenario must be entered in this table. The scenario will reference the KUID by name, which is case-sensitive. For example, when the script loads the map, it will reference it by the name "Tutorial1", which will be looked up in the KUID-table, and found as KUID -2:2023211879. Trains and rolling stock etc are referenced in the same manner. You will also notice that the scenario has its own unique KUID. The description text is displayed in the scenario selection screen.

6. Next, we will begin editing the TrainzScript file. Open the Tutorial1.gs file with your desired programming editor. This file is created from the template.gst in the scripts\ folder when you export the Track Side Object (TSO). The syntax of the following script is explained in the compiler documentation, but if you have a read through it, it is pretty self-explanatory. The file looks like this.

```

include "trainz.gs"

//
// class MyTutorial1
// brief This is the scenario class. Modify
this class with
// your own gameplay.
//
game class MyTutorial1 isclass Scenario
{
    Train myConsist;
    bool scenarioDone = false;

    //
    // Load will be called by Trainz to load the
    scenario map, and when the user presses Ctrl-L
    // param data is the save game data if
    loading a saved game.
    //
    bool Load(string data)
    {
        if(data and data.size())
        {
            Interface.Load(data);
        }

        // load the map
        if(!World.LoadMap(World.
FindKUID("Tutorial1"))
        {
            Interface.Log("Error loading scenario
map");
            return false;
        }

        return true;
    }

    //
    // Save will be called by Trainz when the
    user presses Ctrl-S.
    // return the save game string, such that
    load will be able to restor the save game
    // from the last save check point.
    //
    string Save()
    {
        return Interface.Save();
    }

    //
    // TrainDerailed will be called by Trainz
    when a train derails
    //
    void TrainDerailed(int trainId)
    {
        if(!scenarioDone)
        {
            World.EndScenario(10);
            scenarioDone = true;
        }
    }

    //
    // TrainCollided will be called by Trainz
    when a train collides
    //
    void TrainCollided(int trainId)

```

```

{
    if(!scenarioDone)
    {
        World.EndScenario(10);
        scenarioDone = true;
    }
}

//
// TrainSpeedingFine() is called by Trainz
every second your trains speed exceeds the
floating limit
//
void TrainSpeedingFine()
{
    //Interface.AdjustScore(-10);
}

//
// TrainBadCouple() is called by Trainz when
vehicles couple greater than 8KPH.
//
void TrainBadCouple(int vehicleId)
{
    //Interface.AdjustScore(-200);
}

//
//
// main thread
// brief main is executed automatically after
Load() is called. edit
// main to contain your scenarios gameplay.
//
//

thread void main(void)
{
    // Start the monitor thread to monitor
speeding, derailing etc.
    Monitor();

    //
    // create consist specs
    //

    //
    // create consists
    //

    //
    // gameplay
    //

    scenarioDone = true;
}
};

```

7. The final step is to compile the scenario. Copy the makescript.bat file from Scripts\Docs folder into your World\Custom\Scenarios\Tutorial1 folder. Run this batch file and make sure no errors are reported (consult the forum for assistance). Notice that this batch file uses the TrainzScript compiler (gsc) to create the Tutorial1.gsl file.

8. You are now ready to launch your very first scenario. The scenario should load your Tutorial1 map. You will notice that the camera is focused on the junction we placed on the map. Had we not placed that junction, the map would not be visible, as Trainz has nowhere to focus the camera.

You have now successfully created your very first scenario. Study the Highland Valley scenarios and the script .gs files for further TrainzScript information. Scripting questions may also be asked on the forums.

Good luck, and we hope you will enjoy your scripting efforts

CHAPTER 12

Appendix A - Classes and Codes

CATEGORY CLASS

The Category Class is listed in the config.txt file of each item of content.

They represent a standardized system for referring to the various types of Locos, Rolling stock, Scenery, Spline and Industry assets. Future version of Trainz will also make use of the codes so adding them to any locos and rolling stock you create is advisable.

Categories listed in red are new to TRS

The Category Classes are:

Class "A"	Motive Power
Class "B"	Buildings and Structures
Class "C"	Cabeese
Class "D"	Defence
Class "E"	Environment
Class "F"	Foliage
Class "G"	Ground
Class "H"	Mesh
Class "I"	Product
Class "J"	Texture
Class "L"	Light Rail & Monorail
Class "M"	Maintenance Of Way
Class "O"	Organism
Class "P"	Passenger & Mail Cars
Class "R"	Railcars & Multiple Unit Sets
Class "S"	Splines
Class "T"	Track
Class "V"	Vehicles
Class "W"	Wayside
Class "X"	Freight Cars
Class "Y"	Maps and Scenarios
Class "Z"	Train Parts

Each Category Class may have many subcategories as listed below. Please choose the most appropriate Category Class for your item.

Selecting a correct Category Class is important since Trainz will allow users to use the Category Class as a sort and selection criteria inside My Collection and in Drivers Consist selection screen

A MOTIVE POWER

AA	Electric Multi-current
AC	AC Electric
AD	DC Electric
AE	Experimental or Special
AG	Gas Turbine
AH	Diesel Hydraulic
AL	Diesel & Diesel Electric
AM	Mammal
AS	Steam Loco & Tender
AT	Steam Tank

B BUILDINGS & STRUCTURES

BC	Commercial (<i>scenery non-functional</i>)
BI	Industrial (<i>scenery non-functional</i>)
BH	Home & Residential (<i>scenery non-functional</i>)
BR	Railway (<i>scenery non-functional</i>)
BS	Special (e.g. military) (<i>scenery non-functional</i>)
BT	Traffic & Streetscape (<i>scenery non-functional</i>)
BU	Utility (incl. Civil buildings) (<i>scenery non-functional</i>)
BIN	Industry asset with product processing functionality
BPF	Passenger Station with passenger processing functionality
BPN	Passenger Station (non-functional)
BB	Buildable (Kind Buildable)

C	CABEESE	OH	Human
CB	Brake van	OHD	Locomotive Driver
CC	Caboose		
D	DEFENCE	P	PASSENGER & MAIL CARS
DA	Military motive power	PA	Suburban/short haul (no W.C.)
DE	Military experimental & special vehicles	PB	Baggage cars
DP	Military equipment - lab & personnel vehicles	PC	Coach/chair cars
DX	Military equipment - freight	PD	Dome cars
		PH	Bar/cafeteria cars
E	ENVIRONMENT	PL	Lounge cars
ES	Sky	PM	Mail cars
EW	Water	PO	Observation cars
		PP	Power cars
F	FOLIAGE	PR	Buffet/dining/restaurant cars
FC	Cactii	PS	Sleeping cars
FF	Flowers	PU	Special cars (e.g. Gaming Cars)
FO	Orchards & Crops	PV	Private cars
FS	Shrub	PX	Composite passenger cars
FT	Trees		
G	GROUND	R	RAILCARS & MULTIPLE UNIT SETS
GA	Arid	RA	AC electric
GL	Lush	RC	DC electric
GS	Seasonal	RD	Diesel & diesel electric
		RH	Diesel hydraulic
H	MESH	RP	Petrol
HM	Mesh	RS	Steam
I	PRODUCT	S	SPLINES
IC	Container Category	SF	Fences
IP	Passenger Category	SR	Roads
IB	Bulkload Category	SP	Platforms
IL	Liquid Category	SS	Structure
		SV	Vegetation
J	TEXTURE	T	TRACK
JC	Corona	TB	Bridge
JI	Icon	TR	Rails
JP	Particle Effects Texture	TT	Tunnel
JO	Other Texture	TF	Fixed Track
L	LIGHT RAIL & MONORAIL	V	VEHICLES
LS	Articulated train sets	VA	Air
LT	Trolleys, trams & streetcars	VL	Land
LM	Monorail vehicles	VS	Sea
M	MAINTENANCE OF WAY	W	WAYSIDE
MA	Camp vehicles	WA	Signalling
MB	Ballast cars	WS	Trackside signage
MC	Cranes/lifting	WX	Accessories
MD	Diagnostic vehicles (e.g. dynamometer)	X	FREIGHT CARS
ME	Instructional vehicles	XA	Auto transporter
MF	Fire vehicles	XAA	Open sides
MI	Inspection vehicles	XAB	Auto box car
MT	Track vehicles (e.g. tamper)	XB	Box car/covered van
MP	Snow ploughs	XBD	Dangerous goods
MS	Section cars (e.g. fairmont)	XBG	General service
MX	Freight equipment (for MoW traffic)	XBI	Insulated
MW	Weed spray		
O	ORGANISM	XF	Flat
OA	Animal Kingdom	XFA	Articulated

XFC	Intermodal
XFD	Depressed center
XFH	Heavy duty
XFM	General service
XG	Gondola/open wagon
XGB	Bottom dumping
XGC	Combination bottom/end/side dumping
XGE	End dumping
XGR	Rotary dumping
XGS	Side dumping
XGT	Covered
XH	Hopper
XHB	Bottom dumping
XHC	Combination bottom/end/side dumping
XHE	End dumping
XHR	Rotary dumping
XHS	Side dumping
XHT	Covered
XI	Foundry
XIB	Bottle/torpedo cars
XIT	Tipper/slag cars
XL	Livestock
XLA	Single deck
XLC	Multiple deck and convertible
XLH	Horse box
XR	Refrigerated
XRI	Ice chilled
XRM	Mechanically chilled
XS	Special
XSN	Novelty
XSU	Unclassified
XT	Tanker
XTA	Domeless
XTS	Single dome
XTM	Multiple dome
XV	Ventilated car/louvred van
XVG	General service
XVP	Produce service
Y	MAPS & SCENARIOS
YM	Map
YS	Scenario
YP	Profile/Session
YR	Rule
YD	Driver Command
YH	HTML-Asset
Z	TRAIN PARTS
ZB	Bogie/Truck
ZE	Enginespec
ZH	Hornsound
ZI	Interior
ZP	Pantographs
ZS	Enginesound
ZX	PaintShed-Template

REGION CODES

Region codes are a single or multiple line code that is included in the config.txt file. **The codes are used in Trainz as a sort and selection criteria.** For content that exists in multiple areas, select appropriate codes in CCP.

For example, a locomotive that was available in the United States and Canada would be specified as follows:

```
category-region-0 US
category-region-1 CA
```

The region codes that are recognized by Trainz are as follows:

AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AI	Anguilla
AL	Albania
AM	Armenia
AN	Netherland Antilles
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AZ	Azerbaijan
BA	Bosnia-Herzegovina
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BM	Bermuda
BN	Brunei Darussalam
BO	Bolivia
BR	Brazil
BS	Bahamas
BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos (Keeling) Isl.
CF	Central African Rep.
CG	Congo
CH	Switzerland
CI	Ivory Coast
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China
CO	Colombia

CR	Costa Rica
CS	Czechoslovakia
CU	Cuba
CV	Christmas Island
CY	Cyprus
CZ	Czech Republic
DE	Germany
DJ	Djibouti
DK	Denmark
DM	Dominica
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
EH	Western Sahara
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FK	Falkland Isl.(Malvinas)
FM	Micronesia
FO	Faroe Islands
FR	France
GA	Gabon
GB	Great Britain
GD	Grenada
GE	Georgia
GH	Ghana
GI	Gibraltar
GL	Greenland
GP	Guadeloupe (Fr.)
GQ	Equatorial Guinea
GF	Guyana (Fr.)
GM	Gambia
GN	Guinea
GR	Greece
GT	Guatemala
GU	Guam (US)
GW	Guinea Bissau
GY	Guyana
HK	Hong Kong
HM	Heard & McDonald Isl.
HN	Honduras
HR	Croatia
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IN	India
IO	British Indian O. Terr.
IQ	Iraq
IR	Iran
IS	Iceland
IT	Italy
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kirgistan Ex-USSR
KH	Cambodia
KI	Kiribati

KM	Comoros	QA	Qatar
KN	St.Kitts Nevis Anguilla	RE	Reunion (Fr.)
KP	Korea (North)	RO	Romania
KR	Korea (South)	RU	Russian Federation Ex-USSR
KW	Kuwait	RW	Rwanda
KY	Cayman Islands	SA	Saudi Arabia
KZ	Kazachstan	SB	Solomon Islands
LA	Laos	SC	Seychelles
LB	Lebanon	SD	Sudan
LC	Saint Lucia	SE	Sweden
LI	Liechtenstein	SG	Singapore
LK	Sri Lanka	SH	St. Helena
LR	Liberia	SI	Slovenia
LS	Lesotho	SJ	Svalbard & Jan Mayen Is
LT	Lithuania	SK	Slovak Republic
LU	Luxembourg	SL	Sierra Leone
LV	Latvia	SM	San Marino
LY	Libya	SN	Senegal
MA	Morocco	SO	Somalia
MC	Monaco	SR	Suriname
MD	Moldavia Ex-USSR	ST	St. Tome and Principe
MG	Madagascar	SU	Soviet Union
MH	Marshall Islands	SV	El Salvador
ML	Mali	SY	Syria
MM	Myanmar	SZ	Swaziland
MN	Mongolia	TC	Turks & Caicos Islands
MO	Martinique (Fr.)	TD	Chad
MR	Mauritania	TF	French Southern Terr.
MS	Montserrat	TG	Togo
MT	Malta	TH	Thailand
MU	Mauritius	TJ	Tadjikistan Ex-USSR
MV	Maldives	TK	Tokelau
MW	Malawi	TL	East Timor
MX	Mexico	TM	Turkmenistan Ex-USSR
MY	Malaysia	TN	Tunisia
MZ	Mozambique	TO	Tonga
NA	Namibia	TR	Turkey
NC	New Caledonia (Fr.)	TT	Trinidad & Tobago
NE	Niger	TV	Tuvalu
NF	Norfolk Island	TW	Taiwan
NG	Nigeria	TZ	Tanzania
NI	Nicaragua	UA	Ukraine
NL	Netherlands	UG	Uganda
NO	Norway	UK	United Kingdom
NP	Nepal	UM	US Minor outlying Isl.
NR	Nauru	US	United States
NT	Neutral Zone	UY	Uruguay
NU	Niue	UZ	Uzbekistan Ex-USSR
NZ	New Zealand	VA	Vatican City State
OM	Oman	VC	St.Vincent & Grenadines
PA	Panama	VE	Venezuela
PE	Peru	VG	Virgin Islands (British)
PF	Polynesia (Fr.)	VI	Virgin Islands (US)
PG	Papua New Guinea	VN	Vietnam
PH	Philippines	VU	Vanuatu
PK	Pakistan	WF	Wallis & Futuna Islands
PL	Poland	WS	Samoa
PM	St. Pierre & Miquelon	YE	Yemen
PN	Pitcairn	YU	Yugoslavia
PT	Portugal	ZA	South Africa
PR	Puerto Rico (US)	ZM	Zambia
PW	Palau	ZR	Zaire
PY	Paraguay	ZW	Zimbabwe

ERA CODES

Era codes are used in Trainz as a sort and selection criteria. For content that exists in multiple eras list each era in CCP.

For example, a locomotive that was available in the 1960s and 1970s would be specified as follows:

category-era-0 1960s

category-era-1 1970s

The era codes that are recognized by TRS are as follows:

1800s
1810s
1820s
1830s
1840s
1850s
1860s
1870s
1880s
1890s
1900s
1910s
1920s
1930s
1940s
1950s
1960s
1970s
1980s
1990s
2000s
2010s

Appendix B - Kinds and Containers

This is a tabulation showing the Kinds and their dependant Containers, as a quick reference. Please refer to [Chapter 6](#) for a complete description of the Containers and dependant Tags used in TRS2006. There are too many tags to do a similar tabulation for Containers and tags.

Kinds																								
Containers	MOCrossing	MOJunction	MOSignal	MOSpeedBoard	Paintshed-Template	Paintshed-Skin	Pantograph	Product	Product-Category	Profile	Region	Scenery	Scenery-Trackside	Steam-Engine	Texture	Texture-Group	Track	Tracksound	Traincar	Tunnel	Turntable	Water2		
allowed categories	✓	✓	✓	✓								✓	✓						✓		✓			
allowed products list	✓	✓	✓	✓								✓	✓						✓		✓			
animation effect	✓	✓	✓	✓			✓					✓	✓						✓		✓			
attached track	✓	✓																			✓			
attached trigger	✓	✓																			✓			
attachment effect	✓	✓	✓	✓			✓					✓	✓						✓		✓			
attachment points	✓	✓	✓	✓								✓	✓						✓		✓			
bogeys																			✓					
cameralist																								
conflicts with queues	✓	✓	✓	✓								✓	✓						✓		✓			
consists																								
corona effect	✓	✓	✓	✓			✓					✓	✓						✓		✓			
driver-settings																								
dynamic brake														✓										
effects	✓	✓	✓	✓			✓					✓	✓						✓		✓			
extensions	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
flowsize														✓										
junction-vertices																								
kuid table	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
levels																								
lights			✓																					
mass														✓										
mesh table	✓	✓	✓	✓			✓					✓	✓						✓		✓			
motor														✓										
name effect	✓	✓	✓	✓			✓					✓	✓						✓		✓			
notches														✓										
obsolete-table	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
pressure														✓										
privileges	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
processes																								
queues	✓	✓	✓	✓								✓	✓						✓		✓			
signals			✓																					
smoke	✓	✓	✓	✓								✓	✓						✓		✓			
sound	✓	✓	✓	✓								✓	✓						✓		✓			
soundscript	✓	✓	✓	✓								✓	✓						✓		✓			
steam														✓										
steam power														✓										
string table cn, cz, de, fr, if, pl, ru	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
texture-replacement effect	✓	✓	✓	✓			✓					✓	✓						✓		✓			
textures																✓								
throttle power														✓										
thumbnails	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
tracksound																	✓			✓				
vertices	✓																				✓			
volume																								
world origin																								

ACKNOWLEDGEMENTS

We would like to acknowledge the assistance of a number of members of the community, who have made formal submissions, with comments and suggestions, on the revision of the CCG. They are...

*Andi Smith
Bill Smith
Chris Cordes
Chris Hucklebridge
Chris Margetts
Geoff Steele
Giga Melt "GM"
Jeff Barr
JL Bernard
Martin van Kuilenburg
Mike Banfield
Rhian Geleick
Rick Horwell
Toby Roberts*

...and Eric Whitmore.

This list to be redone following feedback from the darft relese and comments submitted.

Their input has been invaluable and has assisted us to address many of the community questions about content creation.

A general thank you is also extended to the wider Trainz community for their forum posts on content creation issues and suggestions.

Finally, I'd like to thank Ian Manion and Scott Cameron for taking on the revision to suit the new TRS2006 release.

The Brew Crew.
(August 2005)