User's Guide

Release 5

E A I

Sense8products

Copyright

Trademarks

Engineering Animation, Inc.
100 Shoreline, Suite 282
Mill Valley, CA 94941-3645

(415) 339-3200
(415) 339-3201 (fax)

Contents

Preface

This User's Guide is intended to teach you how to use WorldUp to create 3D/VR simulations.

Note All reference material, such as function and object property definitions, is documented in the on-line help.

About this Manual

The following is a brief description of each chapter in this guide:

Chapter 1, *Installing WorldUp*, describes system requirements, installation, and software license configuration for starting and running WorldUp on the Window NT/2000 and Windows 95/98 platforms. Refer to your platform specific Installation Guide for installation instructions for other platforms.

Chapter 2, *Introduction*, provides an overview of the WorldUp product, new features and enhancements in Version 5.0, how to get started in WorldUp, and important contact information.

Chapter 3, *Overview of WorldUp*, introduces you to WorldUp and some of the fundamental concepts that you need to understand before building a WorldUp simulation.

Chapter 4, *Organizing Your Scene*, provides an overview of the Scene Graph and how your entire 3D scene is organized.

Chapter 5, *Working with a Project*, describes how to create a new project (or simulation), open an existing project, and save your project. Also presents information on Directory paths.

Chapter 6, *A Quick Tour*, walks you through four tutorials that acquaint you with the main components of WorldUp and the process of creating a WorldUp simulation. These tutorials are designed to quickly get you started with your own simulations.

Chapter 7, *Using the Workviews*, provides a description of each Workview–Scene, Model, Behavior, and Type–and the Project Workview as a whole. Workviews, new with R5, provide a more efficient workflow by simplifying the User Interface using tabs for the task on which you are focused.

Chapter 8, *Development Window – Navigation and Manipulation*, describes the various navigation and manipulation tools available on the Development Window toolbar and methods for viewing your graphical objects and moving around your scene in the Development Window.

Chapter 9, *Objects and Properties*, defines objects and properties, and describes how to create, delete, duplicate, and find objects.

Chapter 10, *Windows, Viewports, and Viewpoints*, describes the difference between windows, viewpoints, and viewports. Also describes stereo viewing, multi-channel systems, and interleaved display devices.

Chapter 11, *Adding 3D Objects*, describes how to work with primitive objects and import objects from the Model Workview.

Chapter 12, *Editing 3D Objects*, describes how to change the appearance of the objects in your scene.

Chapter 13, *Lights*, describes the effect of lights and materials on a scene.

Chapter 14, *The Behavior System*, describes the system where all the behavior creation and scheduling is done. It describes how to create a new behavior, schedule the behavior, add inputs to the behavior, edit the behavior's properties, and finally to export the behavior for reuse.

Chapter 15, *Paths*, describes the series of position and orientation records that you can use to guide Viewpoint or Movable objects. Also describes how to dynamically create, record, edit, save, load, and play back paths in a variety of ways.

Chapter 16, *Sounds*, describes how WorldUp can play sounds from .WAV files. Scripts control when and for how long a sounds plays.

Chapter 17, *Using Input Devices*, describes how to interface your simulations with each input device supported by WorldUp, such as the Spacetec Spaceball.

Chapter 18, *Multi-User Simulations*, describes how to use WorldUp to create multi-user simulations that can be run with Sense8's World2World server product. If you have not purchased World2World, you will not be able to take advantage of the features described in this chapter.

Chapter 19, *Tips and Tricks*, provides tips on how to optimize your simulations.

Chapter 20, *Publishing Your Application*, provides instructions for packaging and distributing your application within a number of industry standard environments. You can choose from the Web, ActiveX, Visual Basic, MacroMedia Director, Matlab, MS Office, and Visual C++.

Appendix , *Environment Variables*, describes environment variables that you can add to your system to configure your computer for maximum performance.

Appendix , *WorldUp Players and Plug-Ins*, describes the available WorldUp players and plug-ins and describes how to install, distribute, and run them.

Appendix , *WorldUp User's Group*, describes the WorldUp User's Group (SIG-WTK) and how you can join.

Appendix , *WorldUp Shortcuts*, lists various shortcuts that you can use as you develop simulations in WorldUp.

Appendix , *Pre-Built Behavior Library*, lists the various pre-built behaviors WorldUp provides so you can drag and drop them onto objects, rather than write scripts to create them.

Appendix , *WorldUp File Formats*, describes which formats for 3D models WorldUp supports for importing graphical objects into your simulation.

Appendix , *Glossary*, defines many of the common terms used throughout the User's Guide and on-line help, and within the WorldUp development environment.

A detailed index follows the glossary.

Related Documentation

The following publications provide additional information about the WorldUp product:

WorldUp Programmer's Guide – This Guide, also provided with WorldUp R5, provides C and C++ programmers with more detailed information of WorldUp.

WorldUp Plug-In Author's Guide – The Plug-in Kit, provided if you purchase the Plug-In Kit option from Sense8, allows you to quickly create custom simulation objects that directly interface with the WorldUp Object System using a high-level set of object management routines.

WorldUp BasicScript Reference Manual – This manual is your on-line help, and is also found in pdf format on your WorldUp R5 CD. An optional hardcopy is available from EAI/Sense8.

Style Conventions

This manual uses the following conventions:

- `Courier New font` – represents text that you are instructed to type as part of an example or tutorial, or represents script code.
- SMALL CAPS – represents a key on your keyboard, such as SHIFT or ENTER.
- *Italics* – represent emphasized words, chapter titles in cross-references, or the first use of a new term.
- "Quotation marks" – represent chapter section names in cross-references.

1

Installing WorldUp

This chapter describes how to install and run WorldUp on the Microsoft Windows platform. If you are installing on another platform, refer to your platform specific Installation Guide for installation instructions.

System Requirements

WorldUp requires the following minimum configuration (recommendations are also given):

- A PC with a Pentium microprocessor or its equivalent
- At least 16 MB of RAM (24 to 32 MB is recommended for Windows NT)
- A hard drive with 200 MB of free space (typical install)
- A monitor capable of 1024x768 graphics resolution and 16-bit color (recommended)
- A mouse
- A CD-ROM drive (to install WorldUp)
- Windows 98, Windows NT 4.0, or Windows 2000

An OpenGL Graphics accelerator card is *strongly* recommended to speed performance.

You should have at least as much free system RAM as you have dedicated texture memory on your graphics accelerator card. Since OpenGL maintains a copy of all texture maps loaded by your application in virtual memory, you should maximize your virtual memory allocation in Windows so that it can be used to swap texture memory instead of using the hard drive. Swapping to the hard drive will slow your application's performance; so the more RAM you have, the better.

Optional Hardware

You can use WorldUp with just a mouse as an input device or you may want to consider one or more of the devices described below. WorldUp supports a wide range of 2D, 3D, and 6D input sensors, both desktop sensors and sensors worn on the body for sensing position and orientation. In addition to these sensors, WorldUp also supports a standard Windows compatible sound card or system.

WorldUp supports the following types of devices and sensors:

- Mouse (any 2 or 3 button mouse)
- Ascension Mouse
- Ascension Bird and Flock of Birds
- Fifth Dimension Technologies' 5DT Glove
- Gameport Joystick
- General Reality CyberTrak (formerly known as Precision Navigation WayFinder)
- Intersense Trackers
- Logitech 3D Mouse
- Logitech Head Tracker
- Logitech Space Control Mouse (Magellan)
- Polhemus ISOTRAK, ISOTRAK II, InsideTRAK, and FASTRAK
- Spacetec IMC Spaceball 2003 and Spaceball Model 3003 (NT 3.51, using only the pick button)
- StereoGraphics CrystalEyes and CrystalEyes VR LCD shutter glasses
- Thrustmaster Flight Control/Weapons Control Systems and Formula T2 steering console
- VictorMaxx Technologies' CyberMaxx2 HMD
- Virtual i-O i-glasses!

Optional Software

In addition to sensors, you may want to have the following tools handy:

- A 3D modeling tool that generates Autocad DXF, 3D Studio 3DS,Wavefront OBJ, or Virtual Reality Modeling Language (VRML) format files.
- A paint or image processing program capable of generating 24 or 32-bit TARGA format images.
- Sound-capture and processing software or hardware.

Installation Instructions

Follow these steps to install WorldUp on your hard drive:

1. Exit any other applications before starting the installation process.
2. Insert the CD into your CD ROM drive.

   The WorldUp installation process should start automatically. If not, run `setup.exe`.
3. When the Setup program starts, click Next at the Welcome Screen.

4 At the Software License Agreement Screen, read the Software License Agreement and click Yes to accept the terms of the agreement and continue with the installation. Choose No to discontinue the installation.

5 At the Choose Destination Location screen, choose the directory into which you want to install WorldUp. Click Yes to accept the default directory or choose Browse to select a different directory.

   The Setup program will now install the WorldUp directories and files onto your hard drive and then display the Select Groups screen.

6 The Select Groups screen allows you to select a program group that will contain the WorldUp program items. Click Next to accept the default or type in a new program group name to store the program items.

7 At the Finish Installation screen, you can choose to display the `README.TXT` file now by selecting the check box or you can read it later. Click Finish.

The WorldUp installation is complete. Before running WorldUp for the first time, you will need to obtain a WorldUp software license code from SENSE8 by following the directions shown below.

Contents of the WorldUp Installation

Once you have installed WorldUp, you will find that the menu items in the following table have been added to the program group in which you installed WorldUp.

Program Group Item	Description
WorldUp	The WorldUp application.
WorldUp OpenGL Stand-Alone Player	The WorldUp Stand-Alone Player, OpenGL version, is used to view simulations built using WorldUp.
WorldUp D3D Stand-Alone Player	The WorldUp Stand-Alone Player, Direct3D version, is used to view simulations built using WorldUp. You must install DirectX before you can run the Direct3D version of the player. See Appendix B, *WorldUp Players and Plug-Ins* for DirectX installation information.
WorldUp User's Guide	An online version of the WorldUp Users' Guide in Portable Document Format (PDF). PDF files can be viewed using the Adobe Acrobat reader.
WorldUp Online Help	The WorldUp Online Help.
WorldUp License Manager	The WorldUp License Manager is used to display your system-specific ID so that you can obtain a WorldUp software license code that allows you to run WorldUp.

In addition to the program group items, the WorldUp installation adds a number of other files and directories to the WorldUp install directory. The items of special importance are listed in the table below.

File/Directory	Description
readme.txt	Contains last minute changes and known issues concerning your WorldUp software.
\Behaviors	This directory contains the script version of some of the plug-in behaviors for specific use with the script importer and Behavior Wizard.
\Docs	PDF versions of the User's Guide, Programmer's Guide, and BasicScript Reference Manual.
\Drivers	This directory contains device drivers that are either provided by third parties or are simply external to WorldUp.
\Images	This directory contains an assortment of texture image files.
\Models	This directory contains an assortment of geometric model files.
\Plugins	This directory contains the standard plug-ins that ship with WorldUp: **WUPActionSet1.dll** Set of behavior actions plug-in. **WUPTriggerSet1.dll** Set of behavior triggers plug-in. **WUPInterx.dll** Intersense driver plug-in. **WUPInterx.txt** Readme for Intersense driver plug-in.
\Samples	This directory contains subdirectories of various sample simulations which illustrate WorldUp's capabilities. A description of each Samples subdirectory follows: **Samples\Basketball** Shows an interactive Basketball game demo. **Samples\Bond** Illustrates the use of path for motion control. **Samples\Clock** Displays clock time in a 3D environment. **Samples\Dragging** Illustrates dragging in the view plane, and the use of the navigation bar. **Samples\Gears** Illustrates collision detection, gear ratios, dragging, and the use of a navigation bar. **Samples\Grassfire** Complex demo illustrating flight dynamics using a flight dynamics plug-in module **Samples\MarsLander** Illustrates the Behavior System and the use of viewports.

File/Directory	Description
\Samples continued	**Samples\MarsLander** Illustrates the Behavior System and the use of viewports. **Samples\MathEngine** Shows an interface to a WorldUp simulation with a third party library using external DLL calls from script to achieve real-time physics. **Samples\Network** Three versions of a multi-user driving simulator that illustrate the networking features from basic networking to dead-reckoning and chat modes. Requires World2World. **Samples\SQL\Stocks** Illustrates simple database connectivity by accessing a database and translating values into 3D objects **Samples\SQL\ObjectView** Illustrates more advanced database connectivity by reading a list of objects from a database and constructing a scene from those objects. **Samples\Van_Go** A simple driving simulator implemented with scripting.
\Tutorials	This directory contains the project files and the required contents for the Quick Tour chapter of the User's Guide.
\Players	This directory on the WorldUp CD will not be copied onto your system's hard drive during WorldUp installation. It contains a self-extracting file for each of the World Up Players (including the ActiveX Control). These self-extracting executables can be redistributed to other users so they can view WorldUp simulations.
\FloatingLicense Server	This directory on the WorldUp CD will not be copied onto your system's hard drive during WorldUp installation. This directory contains the setup file for the EAI/SENSE8 Products Floating License Server For Windows NT. You should only install this program if you have specifically purchased a floating license agreement.

Getting Started

To get started using WorldUp, you first need to obtain your Software License Code.

Obtaining Your WorldUp Software License Code

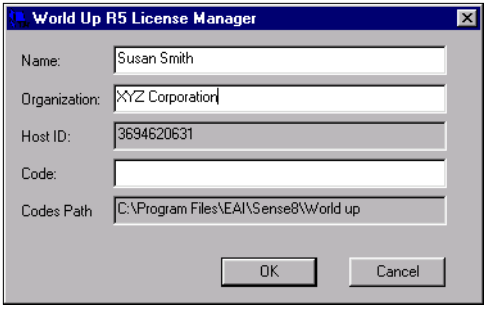
Before running WorldUp for the first time, you need to obtain a unique system-specifc WorldUp Software License Code.

Note If you have obtained a floating or site license, follow the instructions in the documentation provided with those types of licenses instead of the directions shown below.

To obtain your system-specific license code

1 Start the WorldUp License Manager program (license.exe) by clicking its icon in your WorldUp program folder.

The WorldUp R5 License Manager dialog box appears. The system-specific Host ID and the default License Path fields are automatically filled in.



2 Write down your system-specific Host ID:

_____

3 Choose one of these methods (listed in order of quickest response time) to reach SENSE8:

- *World Wide Web* – Point your browser to http://www.sense8.com/licensing/index.html and select WorldUp Code Request. Fill in.the form with your name, company name, the WorldUp Product serial number found on your CD case, the system-specific Host ID (from step 2) and your email address, and submit the information. Your WorldUp License Code will be emailed to you.
- *FAX* – Send a FAX to SENSE8 at (415) 339-3201. Use "Attention: WorldUp License Codes" as the subject of your FAX and include your name, company name, the WorldUp CD's jewel case serial number, and the system-specific Host ID (from step 2). Your system-specific license code(s) will be e-mailed or faxed to you

4 When you have received the system-specific WorldUp Software License Code from SENSE8, start the SENSE8 License Manager again and type it into the Code field of the WorldUp R5 License Manager dialog box and write it down here.

System-Specific License Code:

_____

If you have any trouble obtaining your license code, please send email to Technical Support at support@sense8.com.

5 Click OK.

Once you've entered the system-specific license code into the SENSE8 Product License Information window and clicked OK, the system-specific license code will be saved in a `WTKCODES` file under a designated path (the default path is the directory in which WorldUp was installed), and the path to that file will be placed into your system's registry so WorldUp can find it. You can now run WorldUp by selecting it from your program folder. You do not need to re-enter the system-specific license code every time you start WorldUp, however you will need it if you reinstall WorldUp. For that reason, write down the system-specific license code in the space provided above.

Starting WorldUp

To start WorldUp

1 Click the Start button.

2 Select Programs > WorldUp R5 > WorldUp.

2

Introduction

WorldUp is the next generation 3D content authoring tool for real-time graphical simulations. WorldUp helps you efficiently create or import 3D scenes, make them interactive with easy-to-use drag and drop assembly, and integrate them with the industry standard tools you already use. WorldUp combines several technologies into a tightly integrated, object-oriented environment.

With WorldUp, you create visual simulations in which graphical objects have real world properties and behaviors. You can modify these properties and behaviors while the simulation is running and see immediate results.

WorldUp provides an easy-to-use graphical user interface from which you create objects and properties, and design your simulations. You may also import 3D models from industry standard modelers. To add behaviors to your objects, you can author customer behaviors or change a property of an existing behavior by writing scripts using the BasicScript language, or use property change events to trigger behaviors. And now, in WorldUp Release 5 (R5), you can drag and drop pre-built behaviors onto objects.

To navigate within a WorldUp simulation and control its objects, you use input devices, such as a mouse or Spaceball. You can view the simulation on your computer monitor or in a stereo display device, such as a head-mounted display.

When your simulation is complete, you can distribute it to your end-users using a variety of freely-distributable WorldUp players.

With WorldUp, you can develop a prototype in a fraction of the time it takes to develop one using traditional development libraries. Simulations that once took weeks to build can now be built in hours.

What is 3D Content?

WorldUp is a 3D content authoring tool. In a real-time 3D content application, the content can be thought of as the sum of these parts:

- objects
- organization and relationship
- behavior
- systems management
- human interface controls

For example, if you use WorldUp to create a driving simulator, this is how each part of the simulation would fall into the categories shown above:

- The *objects* are the car, the wheels, multiple sections of road, the sound of the car horn, the viewpoint, and the window containing the viewpoint.
- The *organization and relationship* is how the wheels are attached to the car body.
- The *behavior* is how the car rolls over the road and how the wheels rotate when the car moves.
- *Systems management* refers to which section of road the simulation draws and when it is drawn.
- *Human interface controls* include which mouse button to use for acceleration and how to steer left or right.

Together, all the parts create the 3D content of the simulation. In WorldUp, all of the components of a simulation are stored together in a *project* file that contains the content and also references your model, image, script, and sound files. This project file is the .UP file you create, load, or run with WorldUp.

WorldUp Features

The WorldUp development environment contains many features that are valuable when creating virtual worlds, including:

- A graphical, interactive development environment.
- Built-in object types.
- An object-oriented application framework.
- Object hierarchy with dynamic property inheritance.
- The ability to modify and add object properties and behaviors *during a simulation*.
- Support for a wide variety of input/output devices, including devices supported in WorldToolKit Release 9.
- Support for a variety of file formats.
- An interpreted scripting language (BasicScript) syntactically equivalent to Visual Basic.
- A script debugger to help resolve errors in your scripts.
- Run-time binding of C routines from scripts.
- The Scene Graph pane to visualize complex databases.
- Data visualization through BasicScript SQL functions.

- Full integration with SENSE8's World2World server product, allowing you to easily create multi-user simulations (if you have purchased World2World).
- Players for distributing your application.
- Cross-platform portability (without recompiling).
- Online Help.

What's New in Release 5

WorldUp Release 5:

- Makes it easy to create richer 3D interactive simulations and to integrate them into your environment.
- Offers higher power, reusability, and extensibility to developers familiar with C and C++.
- Provides learning paths to facilitate the shift from novice to expert developer.

Easier to create and integrate richer 3D interactive simulations into your environment.

- New drag and drop 3D simulation assembly for simulation building without scripting (using pre-built objects, behaviors, triggers, and materials).
- More efficient workflow with new Workviews which simplify the user interface using tabs to present the task on which you are focused.
- Better placing and viewing of objects by using the new viewports which provide multiple views of a scene.
- Revamped navigation system–includes shadowing, grid placement, and drop lines to facilitate scene assembly and offer rich feedback during the development process.
- High-performance 3D includes reflection mapping, anti-aliasing, true transparency, and support for multiple OpenGL rendering viewports.
- Integrated, multi-user, collaborative networking support with World2World.
- Extensive 3D file format and I/O device support including a new VRML2 read/write and new Multigen OpenFlight reader and new CAD import-capable Direct Model (.jt) file reader
- New, more thorough documentation which now includes a User's Guide and a Programmer's Guide.

Higher power, reusability, and extensibility to developers familiar with C and C++.

- Seamless web-enabled delivery and ActiveX integration by offering official support for simulation-building containers, such as Mathworks MATLAB/Simulink, Microsoft's Visual C++ 6, Macromedia's Director, Visual Basic 6, Netscape, Internet Explorer, Excel, Access, and MS Office.
- Offers an Optional Powerful Plug-in Kit for integrating 3rd party APIs, legacy C code, or extending pre-built behaviors and objects, device support, special effects and OpenGL code.
- Reusable plug-ins for rapid simulation assembly.

Learning paths facilitate the shift from novice to expert developer.

- A new Behavior Wizard for rapid and easier script creation. The Behavior Wizard generates boiler-plate code required to create behaviors and triggers, and places the code into the appropriate place in the simulation system.
- Sample scripts and plug-ins are also included to facilitate learning.

Technical Support

If you have a question about WorldUp, first look for the solution in this manual or in the online Help.

To start the online help in WorldUp

➤ Select WorldUp Contents from the Help menu.

If you cannot find the answer in the documentation or online Help, you can generally save yourself time and effort by reading the WorldUp support page on the SENSE8 web site. It contains answers to Frequently Asked Questions (FAQs) about WorldUp and may also contain links to download the latest WorldUp patches. The web site also contains a knowledge repository.

If you cannot find the answer to your question on the SENSE8 web site, post your question on `sig-wtk@sense8.com`, where the user-base is very helpful and efficient in answering your questions.

If you cannot get the answer to your questions in any of these areas, please contact SENSE8 Technical Support using one of the following methods:

Technical Support on the Web

www.sense8.com/support/index.html

Technical Support by Email

support@sense8.com

Technical Support by Fax

(415) 339-3201

Technical Support by Telephone

(415) 339-3392

WorldUp License Request Web Site

www.sense8.com/support/index.html

WorldUp License Request by Fax

(415) 339-3201

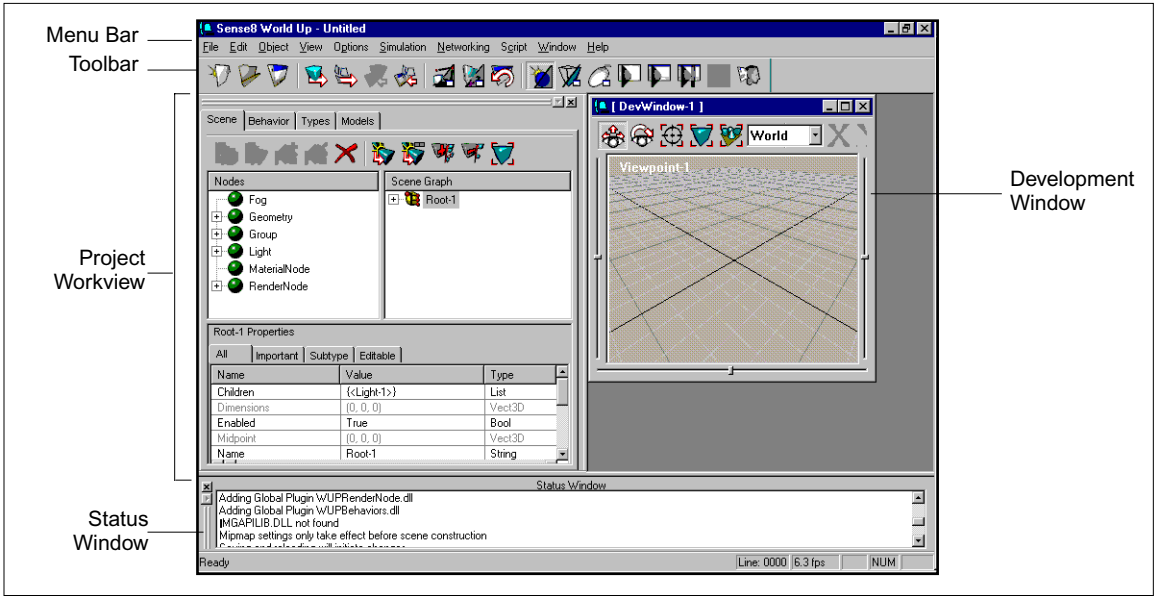
3

Overview of WorldUp

This chapter introduces you to the WorldUp development environment and to some of the fundamental concepts that you need to understand before building a WorldUp simulation.

The WorldUp Window

When you first start WorldUp, the WorldUp window opens with a new, blank project.



The WorldUp Window

Menu Bar

The WorldUp menu bar contains all of the commands available to WorldUp. Many of these commands can also be accessed from the toolbars or by right-clicking when an item is selected.

Toolbar

The main toolbar provides quick access to a number of the WorldUp commands. The Development window and Workviews also contain task-specific toolbars.

Project Workview

The Project Workview is the main interface to the simulation you are building. The tasks you perform in the Project Workview are divided into four tabbed Workviews: Scene, Behavior, Type, and Models.

You access these Workviews by selecting the desired tab. Each of the Workviews is described in detail in Chapter 7, *Using the Workviews*.

Status Window

The Status window is used by WorldUp to display informational status or error messages.

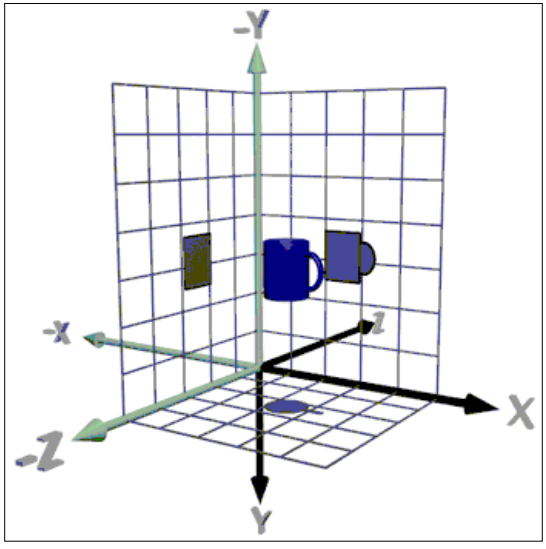
The Development Window

The Development window is where your scene is rendered. For more information about this window, see Chapter 8, *Development Window – Navigation and Manipulation*.

Understanding Real-Time Simulations

Every aspect of the simulation takes place within a *universe*. A universe is the finite volume of space in which all graphical objects appear in a simulation. Any location in space can be described by its *x,y,z* coordinates relative to the origin. The *origin* is the center of the universe, which is located at *x,y,z* coordinates 0,0,0.

In a three-dimensional (3D) coordinate system, the *x* coordinate refers to the left-right location, the *y* coordinate refers to the up-down location, and the *z* coordinate refers to the near-far location. The following illustration shows an object and the *x, y,* and *z* axes using WorldUp's orientation.



The following table shows the direction in which an object moves relative to the coordinate axes, depending on whether you use positive or negative values for the coordinates.

Coordinate	Direction
+ x	Right
- x	Left
+ y	Down
- y	Up
+ z	Forward
- z	Back

The movement of an object from one (*x,y,z*) location to another is known as *translation*. The *rotation* of an object is described in three ways:

- *Pitch* – the rotation of an object about its *x* axis.
- *Yaw* – the rotation of an object about its *y* axis.
- *Roll* – the rotation of an object about its *z* axis.

Coordinate Systems

The concept of a *coordinate system* is fundamental to real-time 3D graphics and to WorldUp.

A coordinate system (sometimes called *reference frame* or *context*) refers to the *x*, *y*, and *z* coordinate axes used to describe position and orientation in space.

The Scene Workview provides a way of assembling objects hierarchically, so that the location of any object is relative to the coordinate system of its parent in the Scene Workview. Let's say you have an object called Road and an object called Car. If you nest Car underneath Road in the Scene

Workview, then the translation property of Car specifies the position of the car in the reference frame of (or in relation to) the road.

Reference frames that are commonly referred to in WorldUp are described below.

World

The World (or Global) reference frame is independent of the objects in the universe, and is fixed in space. The World coordinate system originates at the center of the universe, defined as *x,y,z* coordinates 0,0,0.

Local

The Local reference frame is specific to each Node object (that is, any object that can appear in the Scene Workview). For a geometry, it is the reference frame in which the geometry was modeled. Let's say you have a geometry representing an airplane that was modeled with the *z*-axis coinciding with the length of the airplane. If you rotate the airplane about its *z*-axis in its local reference frame, for example, by using the Roll function in a script, it will roll about its length, regardless of the airplane's orientation in space.

Parent

The reference frame of an object's parent in the Scene Workview.

Viewpoint

The coordinate system defined by the position and orientation properties of a given Viewpoint object. This reference frame can be useful, for example, when positioning geometries (see "Controlling the Scene Graph Dragging Options" on page 29 for more information).

The Building Blocks

This section describes the WorldUp building blocks, the simulations components that you use to build a real-time simulation. These building blocks include objects, object types, properties, the scene graph, and behaviors.

Object Types, Objects, and Properties

Object *types* are displayed in a hierarchical structure in the Type Workview. These types consist of *properties* that describe the characteristics of an object. You use object types as templates for creating subtypes or objects, which then inherit all of the properties from their type.

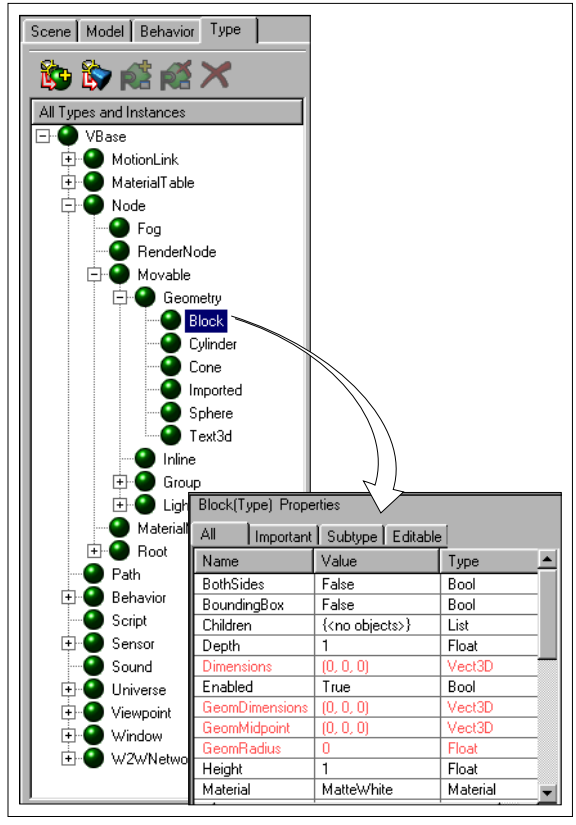
A *subtype* is any type that is the child of another type. Since all types are ultimately derived from the VBase object type, all types except VBase are also considered subtypes. Thus, the two terms are often used interchangeably.

WorldUp Object Types

Type	Description
Motion Link	Links a viewpoint or geometry to a Path or sensor for motion control.
Node	The base type for all objects that participate in the scene graph (see Chapter 4).
Path	A collection of position and orientation elements used to animate objects (see Chapter 15).
Behavior	The base type for Triggers and Actions, which handle events and modify objects (see Chapter 14).

Type	Description
Script	A text file containing BasicScript language code which controls a simulation (see the *WorldUp Programmer's Manual* for more information).
Sensor	The base type for all devices supported by WorldUp (see Chapter 17).
Sound	A sound file that can be attached to objects to achieve 3D spatialized sound (see Chapter 16).
Universe	The global simulation object that stores simulation level preferences (see Chapter 5).
Viewpoint	A representation of the user's perspective used to calculate how objects are drawn in the 3D world (see Chapter 10).
Window	The frame in which the simulation is rendered (see Chapter 10).
W2Wnetwork	The base networking component that establishes and determines a simulation's connection with a World2World server (see Chapter 18).

The following figure shows the Block object type in the Type Workview and its corresponding properties in the Property pane. Any subtype or object that is created from Block will include all of Block's properties.



Block object type and its Property pane

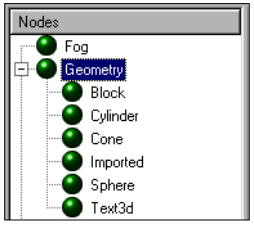
WorldUp comes with several pre-defined types. In the Type Workview, green spheres represent pre-defined types, and green spheres with a yellow plus represent user-defined types.

Properties can only be added to user-defined subtypes. That is, you cannot add properties to WorldUp default object types. This preserves the integrity of the default types as building blocks for all your simulations. Nor can you add properties directly to objects. Thus, if you are creating an object that requires a property that does not already exist under the WorldUp object types, you need to create a subtype and add the necessary properties.

Subtypes are also useful if you plan to create a number of objects that require similar property values. For example, let's say you want to create a variety of Block objects. They may vary in height and color, but you want them all to have a depth of 3 and a width of 4. You could create a subtype of Block and set its Depth property to 3 and its Width property to 4. Whenever you need to create a new block, create it from this subtype.

Graphical Objects

Objects that you can see in the Development window are referred to as *graphical* objects. These are objects that are created from subtypes of the Geometry type in the Type Workview (the Block, Cylinder, Cone, Imported, Sphere, and Text3d types as shown in the figure below).



Other types of objects may be in your simulation, such as Lights or Switchers, but they are not visible.

Graphical objects are one of the building blocks of a simulation. A graphical object is a discrete, movable object such as a wheel, and is composed of one or more polygons, each of which contains three or more vertices. Examples of graphical objects are balls, vehicles, cylinders, wheels, houses, and landscapes.

You can interact with graphical objects and they can interact with each other. You can organize graphical objects hierarchically, use sensor devices to affect their motion or state, and write scripts to define their behavior in the simulation.

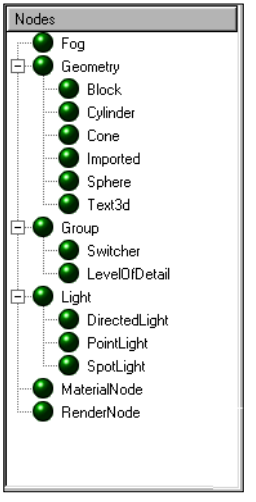
For information on Sensors, see Chapter 17, *Using Input Devices*.

Graphical objects do not obey any default physical laws. If you want a graphical object to fall when you release it, you must assign a task to the object. You do this by either using an existing script or by writing your own script and then attaching it to your object as a task.

For information on scripts, see the *WorldUp Programmer's Guide*.

The Scene Graph

Objects that are created from the Node type or any of its subtypes are referred to as *nodes*. The Nodes pane in the Scene Workview displays all the Node types in WorldUp and their instances. The figure below shows all of the WorldUp default subtypes for the Node type.



The three categories of nodes are graphical, attribute, and organizational. The table below describes each category and the associated object types.

Node Category	Description	Object Types
Graphical	Objects that you can see in the Development window.	Block Cylinder Cone Imported RenderNode Sphere Text3d
Attribute	Objects that affect the appearance of graphical objects.	Fog Light MaterialNode
Organizational	Objects that affect how WorldUp traverses the scene graph.	Group Switcher LevelOfDetail

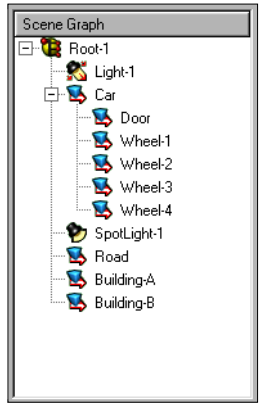
The spatial organization and relationship of Node objects to each other is controlled by your *scene graph*. A scene graph is a hierarchical arrangement of nodes, organized beneath a single Root node. In WorldUp, you view and modify your scene graph using the Scene Graph pane.

The order in which nodes appear in the Scene Graph pane determines the order in which nodes are processed and the order in which graphical nodes are rendered. For each frame of the simulation, WorldUp begins processing the nodes in a top-to-bottom order (as displayed in the Scene Graph pane).

The scene graph enhances performance of the rendering stage (drawing the scene) because it facilitates spatial culling of the scene. In other words, WorldUp calculates which parts of the scene (or scene graph) are visible from the current viewpoint, and quickly rejects non-visible geometry before drawing begins.

The structure of your scene graph determines the parent/child relationships of the objects in the scene. For example, if you nest Node B beneath Node A, B becomes a child of A. A child always translates (moves) and rotates with its parent.

Observe the following scene graph:



In this example, Door, Wheel-1, Wheel-2, Wheel-3, and Wheel-4 are children of Car. If you move Car in the Development window, its children move with it. Light-1, positioned directly beneath Root-1, illuminates all the graphical objects in the scene since it is processed first. Spotlight-1, positioned below Car and its children, illuminates Road, Building-A, and Building-B, but does not illuminate Car and its children since they have already been rendered before Spotlight-1 takes effect.

Notice, however, that because Spotlight-1 is not a child of Car, it does not move with Car. Attribute nodes only affect nodes that are at the same hierarchical level beneath it, or are nested within it. So, if Spotlight-1 was a child of Car, it would not

illuminate the road or the buildings. To cause Spotlight-1 to move with Car, you would add behaviors, as described below.

Behaviors

Behaviors are activities applied to or demonstrated by any WorldUp object within the simulation. This activity is defined by the functions of one or more Trigger and/or Action objects. Triggers and actions are assembled together to form higher-level behaviors. For example, a reflex behavior is witnessed as the result of an event (tapping the knee with a hammer) and a response (the calf flexing upwards).

WorldUp ships with a basic library of Triggers and Actions. In addition to these, you can author your own behaviors, either in script using the Behavior Wizard or in C/C++ using the WorldUp Plug-in Kit (available as a separate module). You can also download behaviors authored by other users from the Sense8 Online Repository.

WorldUp also provides an intuitive drag-and-drop Behavior System for assembling behaviors into higher-level event networks that allow you to control your simulation without programming.

For more informaton on behaviors, see Chapter 14, *The Behavior System*.

Scripts

In WorldUp, a *script* is a text file containing code written in the BasicScript language, which is syntactically equivalent to Visual Basic. A script contains one or more routines that define the behavior to add to an object, such as making an object rotate 10 degrees along its *y*-axis each frame that the simulation is run.

For information on scripts, see the *WorldUp Programmer's Guide*.

The routines that you use to write scripts for your WorldUp simulations consist of BasicScript routines and WorldUp routines. All routines are documented in WorldUp's online Help. On the Help menu, select Script Reference for descriptions of BasicScript routines, or select WorldUp Commands and Functions for descriptions of routines supplied by WorldUp.

With the right script, you can animate objects, assign motion links, detect collisions between objects, and do much more in your simulations.

The two kinds of scripts are Stand-Alone and Task. Stand-Alone scripts are not attached to any particular object and can be run independent of the simulation. For example, you could write a Stand-Alone script that resets certain property values for various objects. Task scripts are written to work with a particular object in your simulation. Task scripts run repeatedly each frame of the simulation. Once you have written the script, you must attach it to the appropriate object in order for it to work.

Events

A change to a property's value is known as an *event*. You can react to specific property events by specifying an entry point in a script that is to be run each time the event occurs, by routing the property's new value to other properties each time the event occurs, or by routing the property change event to fire a Trigger in the Behavior System using the Property Change Trigger.

Note If you have purchased Sense8's World2World server product, sharing events is what allows multiple users to see each others changes.

For information on how to work with events, see the *WorldUp Programmer's Guide*.

How the Pieces All Fit Together

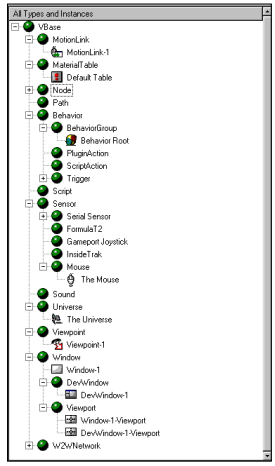
Before you can develop and deploy a WorldUp simulation, the following objects must exist in your project:

- a *Universe* object, containing properties that affect the entire simulation
- an *application window*, in which your simulation displays at run-time
- a *Development window*, in which your simulation displays during development
- a *viewpoint* that specifies how the end-user will view the simulation (all Development windows are associated with a viewpoint; this viewpoint can differ for each window)
- a *mouse sensor*, to manipulate objects in the simulation or to navigate your viewpoint (a mouse is required for development; you can add other sensors to use in addition to the mouse for both development and run-time)
- a *Root object*, which acts as the parent of all objects in the scene graph

The following objects are not required, but you would rarely begin building a simulation without them:

- a *motion link*, which defines the link between the sensor and the viewpoint
- a *Light object*, to illuminate your scene

In anticipation of these requirements, WorldUp provides the following default objects every time you create a new project.



WorldUp Default Objects

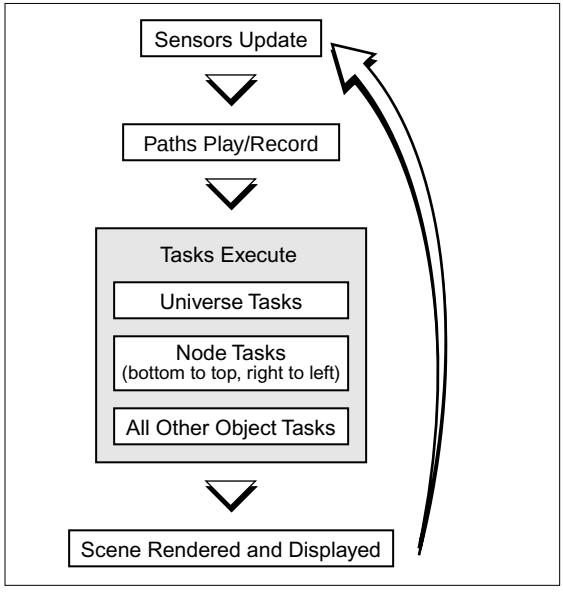
With this foundation, you can then add additional objects from the types previously described, and from the remaining WorldUp pre-defined types described below to form your simulation:

- Node objects, which include the graphical objects that display in the Development window and the objects used to manipulate them.
- Paths, to add movement along a recorded path to specified objects.
- Behaviors, to add custom or pre-built behaviors to your objects, such as causing a ball to bounce.
- Scripts, to add custom behaviors (which are not already pre-built) to your objects.
- Sounds, to add sound to your simulation as it runs.
- W2WConnection and W2WSharedGroup objects, to create multi-user World2World-compliant simulations (only applicable if you have purchased SENSE8's World2World server product). See Chapter 18, *Multi-User Simulations* for more information.

For a complete reference of all WorldUp object types and their properties, see the online Help.

Running Simulations

When you run your simulation, WorldUp goes through the entire simulation loop each frame that the simulation is run. The following illustration shows what happens (and in what order) during a single simulation loop.



Single Simulation Loop

Every object has a list of tasks. These tasks will be executed every frame just before rendering. The following rules apply:

- The tasks on the Universe object will be executed first.
- Tasks on Nodes will be executed next (see "Order of Tasks Within Nodes" below for further details).
- Tasks on the non-Node, non-Universe objects (Windows, Viewpoints, etc.) will be executed last.

Order of Tasks Within Nodes

Within nodes, children's tasks will always be executed before their parent's, and siblings to the right will be executed before siblings to the left.

In short, the order in which objects appear in the scene graph is the reverse order in which their tasks will be executed (execution will occur from the bottom-up in the scene graph). Thus, the Root's tasks will always be the last tasks executed of all the nodes' tasks.

Note If a node is disabled (its Enabled property is set to False), its tasks and the tasks of all of its children will not run. For children under a Switcher node, only those children indicated by the Active Child property will execute. The tasks for all children of LevelOfDetail nodes will be executed.

Starting and Stopping the Simulation

When you are ready to run your simulation, you can run it in the development environment, or you can run it as an application.

- *Running in the development environment* – WorldUp displays the simulation in the window that you have defined as your Development window (see "Creating a Window" on page 97), and allows you to continue to manipulate objects as the simulation runs.
- *Running as an application* – WorldUp displays the simulation in the window(s) that you have defined as your application windows (see "Creating a Window" on page 97). This is a true representation of how the simulation will appear,

and how end-users will be allowed to interact with the simulation when they run it with the WorldUp Players.

Note You can adjust the size of your window as the simulation runs, but be aware that the larger the window, the slower the simulation will run.

Keep in mind that simulations run significantly slower from the WorldUp Development window (even when run as an application) than they do from the WorldUp Players. Thus, as you develop your simulations, it is a good idea to periodically run the simulation from one of the WorldUp Players and take note of the performance differences. For information on running applications from a WorldUp Player, see Appendix B, *WorldUp Players and Plug-Ins*.

To run a simulation as an application

1 Click the Run in AppWindow button on the main toolbar.

If your simulation contains multiple application windows, they will display in the same location unless you have modified the Left Edge and Top Edge properties for the windows.

To see each window, click in the title bar of the application window that is currently in view and drag it away to reveal the application window underneath.

2 To stop the simulation, do one of the following:

- Close the application window(s).
- Click the Stop button.
- On the Simulation menu, select Stop.

To run a simulation in the development environment

1 Click the Run in DevWindow button.

2 To stop the simulation, click the Stop button.

Note Closing a Development window does not stop the simulation; it deletes the DevWindow object.

To run the simulation one frame at a time

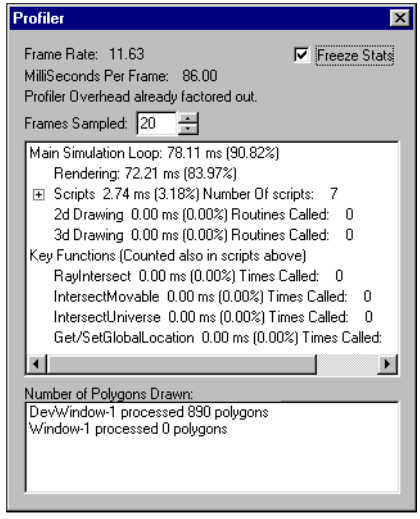
1 Click the Step in DevWindow button.

2 Repeat the Step command to view the simulation frame-by-frame.

Reviewing the Simulation Performance

Reviewing how your simulation uses resources can help you determine how to improve the performance of your simulation.

WorldUp's Profiler (as shown below) provides a variety of statistics about the performance of your simulation, both while the simulation is running or while it is stopped.



At the top of the Profiler, you can see the readings for the simulation's overall frame rate and milliseconds per frame. The Frames Sampled box indicates how many frames prior to the current

frame will be averaged to calculate all readings. The statistics themselves are updated every frame of the simulation.

For example, if the Frames Sampled value is set to 8 at frame 20 of the simulation, the statistics displayed by the Profiler will be an average of the statistics from frames 13 through 20. This can provide more realistic results as performance fluctuates from frame to frame.

However, be aware that increasing the Frames Sampled value to too high a value can also provide misleading results. Suppose something happens in your simulation as it runs that results in a drop in performance (for example, a RayIntersect function may be called). If the Frames Sampled value is very low, you will be able to see a noticeable increase in the statistics as the RayIntersect function is called. If the Frames Sampled value is very high, you will not see that jump in the statistics as it will be averaged out with frames where the RayIntersect function was not being called.

The box in the middle of the Profiler indicates how much of the system's total resources are being used by the simulation, and how much is attributed to specific processes of the simulation loop.

The first line, Main Simulation Loop, indicates the percentage of the system's total resources that are being used by the entire simulation. In the example, this is approximately 90%. Thus, the other 10% of the system's resources are being used by other applications, such as WorldUp and the operating system.

The statistics for Rendering, Scripts, 3D Drawing, and 2D Drawing reflect how much of the system's total resources are being used for those specific processes of the simulation loop. When the simulation is running, the Scripts line is further broken down by individual scripts.

The Key Functions section provides an alternative breakdown of the Scripts statistics. This section shows the amount of resources being used by some of WorldUp's more expensive functions. The statistics shown for these functions have already been factored into the statistics for Scripts.

The box at the bottom of the Profiler indicates the number of polygons that are being rendered for each application window and each Development window in the simulation.

To access and work with the Profiler

➤ Select Profiler from the Simulation menu.

To freeze the statistics for the current frame

➤ Check the Freeze Stats box.

To set the Frames Sampled value

➤ Type a value in the Frames Sampled box, or click the up and down arrows to the right of the value.

To see a breakdown of the resources being used for individual scripts

➤ While the simulation is running, click the plus icon to the left of the Scripts line.

4

Organizing Your Scene

The Scene Graph

The scene graph is the graphical interface that organizes all the objects of a scene and their relationship to each other in a hierarchical tree structure. It's like a family tree where the root is on the top and the branches and leaves extend toward the bottom. A scene is the complete 3D description of everything needed to render an image.

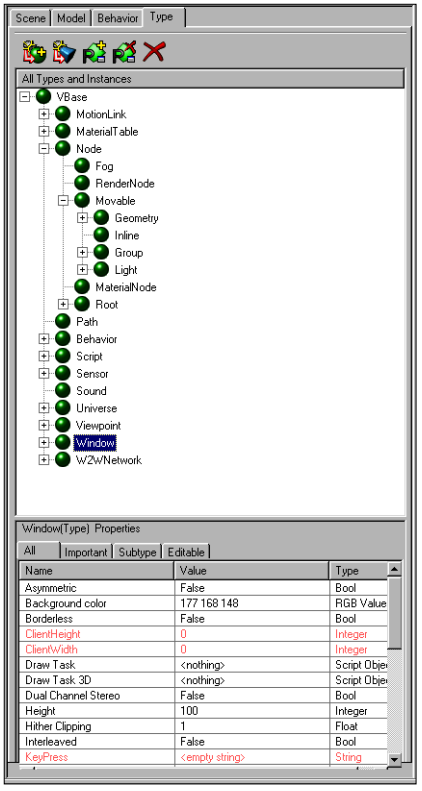
The scene graph is used to:

- establish relationships with objects, such as parent/child
- establish groupings of objects, through Group, Switcher, and LevelOfDetail nodes
- track and select objects
- maintain an overall sense of order

The scene graph hierarchically orders the elements of the scene, known as nodes. A node can hold Graphical objects, Light objects, Fog objects, RenderNode objectss, Material objects, or structural objects used to maintain the hierarchy of the scene graph.

Nodes and Scene Graphs

Objects that are created from subtypes of the Node type are referred to as *nodes*. You cannot create objects or object types directly from the Node object type. It exists to provide its subtypes with the necessary properties and to maintain the coherency of the Type Workview's structure.



Type Workview

Nodes can be categorized as follows:

Node Category	Description	Object Types
Graphical	Objects that you can see in the Development window.	Block Cylinder Cone Imported RenderNode Sphere Text3d
Attribute	Objects that affect the appearance of graphical objects.	Fog Light MaterialNode
Organizational	Objects that affect how WorldUp traverses the scene graph.	Group Switcher LevelOfDetail

The Graphical object types are described in more detail in Chapter 11, *Adding 3D Objects*. Lights are described in Chapter 13, *Lights*. Organizational nodes are described in "Organizational Nodes" on page 25.

The direct subtypes of the Node object type are:

- *Fog* – Fog objects are used to obscure distant objects in the simulation with the color of fog you select. For best results, the Fog color should match the window's background color. Search on Fog type in the online Help for descriptions of the Fog properties.
- *RenderNode*- - RenderNode objects allow you to create your own custom RenderNode type with the optional Plug-in Kit from Sense8. The RenderNode is a node in the scene graph that calls back to a user-defined function in every frame during traversal of the scene graph. The user-defined function contains low-level drawing commands you can execute, allowing you a greater flexibility than is offered by the other physical objects in the scene graph. The benefit

of RenderNode existing in the scene graph is that the node can accumulate the state of the scene graph, including lighting and transformations.

For more details on RenderNode, see the *WorldUp Programmer's Guide*.

- *Movable* – Movable objects can be selected, moved, and rotated in the Development window. For details on the Movable object type and its subtypes, see Chapter 12, *Editing 3D Objects*.
- *MaterialNode* – Using MaterialNode is the most flexible way to add material to an object. MaterialNode allows materials to be created, edited, and saved for reuse in the development environment. For more information, see Chapter 12, *Editing 3D Objects*.
- *Root* – All WorldUp simulations contain a single Root object, which acts as the starting point for your scene graph. Since WorldUp provides you with the necessary Root object (Root-1) by default, you cannot create objects or object types from the Root object type. The Root type exists only to provide you with access to its properties. Search on Root type in the online Help for descriptions of the Root properties.

Organizational Nodes

The spatial organization and relationship of Node objects to each other is controlled by your scene graph. A scene graph is a hierarchical arrangement of nodes, organized beneath a single Root node. In WorldUp, you view and modify your scene graph with the Scene Workview. Understanding the effect of the hierarchical order of your nodes is crucial to creating a successful simulation.

Groups

Objects that you create from the Group type or either of its subtypes (LevelOfDetail and Switcher) are non-graphical, Movable objects that act as containers for other nodes. You specify which nodes you want to be contained in the Group node by making them children of the Group node in the scene graph. See "Working with Scene Graphs" on page 29. You can then use the Group node to manipulate all of its child nodes at once.

Note Information on how to translate and rotate Movable objects, such as Groups, is described in Chapter 12, *Editing 3D Objects*.

The three object types from which you can create Group nodes are:

- *Group* – Objects created directly from the Group type exist simply to act as invisible containers for other nodes in the scene, allowing you to manipulate all of its children at once, such as translating the children together.
- *LevelOfDetail* – LevelOfDetail nodes allow you to specify which child node to render based on its distance from the viewpoint.
- *Switcher* – Switcher nodes allow you to specify whether one, all, or none of its children are rendered at any one time.

Group Nodes

The Group object type contains no unique properties in addition to those provided by the Movable object type. Objects created directly from the Group type exist simply to act as invisible containers for other nodes in the scene graph.

To create a Group node and define its children

1 In the Nodes pane of the Scene Workview, select the Group object type and click the Instantiate Selected Type button.

2 In the Scene Graph pane, press the CTRL key and click on each node that you want to make a child of the Group node.

3 Drag the selection set onto the Group node.

The children of your Group node can now work together as one object as well as independently as separate objects.

LevelOfDetail Nodes

LevelOfDetail nodes are Group nodes that allow you to improve rendering speed by displaying simpler objects at a distance and switching to more complex objects as your viewpoint approaches them in the simulation. It does this by rendering only one of its child nodes at a time. The node that is rendered depends on the distance between the nodes and the viewpoint.

For example, suppose you have a model of a tree that is composed of 50 polygons. You could create two more models that are less detailed: one that is composed of 20 polygons, and one that is composed of 5 polygons. Import these models into your simulation and set them all to the same translation. Make each model a child of a LevelOfDetail node and set distance ranges so that the most detailed model of the tree is rendered when your viewpoint is very close to the object and the least detailed model is rendered when the viewpoint is within the farthest specified range from the object.

Note For tips on creating successful versions of your model to represent each level, see "Model Tricks" on page 197.

To create a LevelOfDetail node and define its children

1 In the Nodes pane, select the LevelOfDetail object type.

2 Click the Instantiate Selected Type button.

Remember that non-imported Movable objects are created at the center of the universe. Since WorldUp will be determining which node to render based on the distance between the viewpoint and the LevelOfDetail node, you will most likely want the LevelOfDetail node to be translated to the same location as its children.

3 In the Scene Graph pane, click one of the nodes that you want to be a child of the LevelOfDetail node.

Note Typically, these nodes are all positioned at the same global location.

4 In the Property pane, single-click the Translation property to position the LevelOfDetail node at the same location as the intended child node.

5 Right-click and select Copy to copy the Translation value.

6 In the Scene Graph pane, select the LevelOfDetail node.

7 In the Property pane, right-click on the Translation property and select Paste to paste the copied text.

8 Press ENTER.

9 In the Scene Graph pane, press the CTRL key while selecting each node that you want to make children of the LevelOfDetail node.

If you want the selected nodes to maintain their global positions when you drag them in the scene graph, select Scene Graph Dragging from the Options menu and ensure that the Scene

Graph Dragging option is set to Objects Preserve Global Position. See "Controlling the Scene Graph Dragging Options" on page 29.

10 Drag the selection set onto the LevelOfDetail node.

11 Arrange the children of the LevelOfDetail node in a descending order so that the node that you want to be rendered when the viewpoint is closest (typically your most complex geometry) is the first child beneath the LevelOfDetail node, and the node that you want to be rendered when the viewpoint is farthest away (typically your simplest geometry) is the last child.

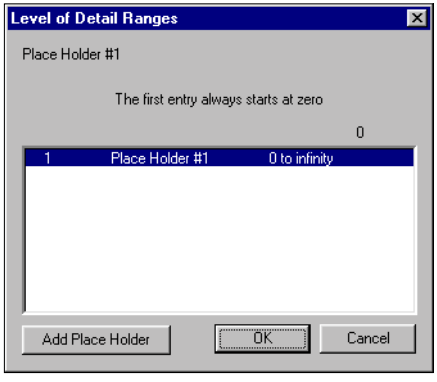
Now you're ready to set the ranges at which the LevelOfDetail node will render each child.

To set ranges for the LevelOfDetail node

1 In the Scene Graph pane, click the LevelOfDetail node.

2 In the Property pane, double-click the Ranges property.

The Level Of Detail Ranges dialog box displays.



Note WorldUp automatically creates a Range entry for each child of the selected LevelOfDetail node. Ranges represent the minimum and maximum number of units from the LevelOfDetail node that the viewpoint must be in order to render the associated node.

The range of the first entry is already set from 0 to infinity. You cannot modify the range of the first entry. The beginning of the range will always be set to 0, and the end of the range will automatically be set to the beginning value that you specify for the next range.

3 Select the second entry.

4 Drag the slider or type a value in the text box at the top to represent the beginning of the next range.

5 Repeat this procedure to set the beginning values for any remaining entries.

6 If you intend to add more child objects to the LevelOfDetail node later and want to specify its range now, click the Add Placeholder button.

A new range entry appears. You can set the range for this entry now, and it will not be used until another child is added to the LevelOfDetail node.

7 Click OK when you are finished.

You can also specify ranges by selecting the Ranges property, clicking on it again for the edit box, typing in a value and pressing ENTER. For example, a value of (20, 40) would set three ranges: 0 to 20 for the first child node, 20 to 40 for the second child node, and 40 to infinity for the third child node.

Switcher Nodes

Switcher nodes are Group nodes that allow you to indicate whether WorldUp will display one, all, or none of the Switcher node's children at any given time.

Switcher nodes are useful for animation effects. For example, suppose you want to create a clown face that changes expression. You could create three different clown models, one that is smiling, one that is frowning, and one that is sad. Import these models into the simulation and make them children of a Switcher node. Write a script that will change the Active Child property for the Switcher object to the geometry that you want rendered as appropriate.

Switchers are also useful for swapping scene pieces, such as different floors in a building. Since you can see only one floor at a time, placing the different floors under a single Switcher node provides an easy way to manage the scene.

To create a Switcher node and define its children

1 In the Nodes pane of the Scene Workview, select the Switcher object type.

2 Click the Instantiate Selected Object button.

3 In the Scene Graph pane, press the CTRL key and click on each node that you want to make a child of the Switcher node.

If you want the selected nodes to maintain their global positions when you drag them in the scene graph, select Scene Graph Dragging from the Options menu and ensure that the Scene Graph Dragging option is set to Objects Preserve Global Position.

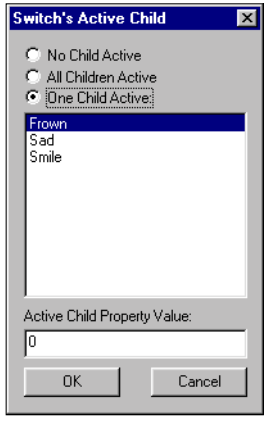
4 Drag the selection set onto the Switcher node.

To specify which Switcher children to render

1 In the Scene Graph pane, select the Switcher node.

2 In the Property pane, double-click the Active Child property.

The Switch's Active Child dialog box displays.



3 Select one of the following settings and click OK:

- *No Child Active* – None of the Switcher's child objects are rendered. This option translates to a property value of `-1`.
- *All Children Active* – All of the Switcher's child objects are rendered. This option translates to a property value of `-3`.
- *One Child Active* – Only one of the Switcher's child objects is rendered. If you click this option, select which child object to make active from the list box in the middle of the dialog box. This option translates to a property value of `0` for the first child, `1` for the second child, `2` for the third, and so on.

You can also specify the Active Child by selecting the Active Child property, typing the appropriate integer (as described above), and pressing ENTER.

To achieve the effect of animation, you could write scripts that change the Active Child value as the simulation runs. To see an example of this, open

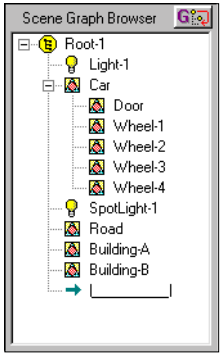
`spinbox.up` in the `samples\switcher` subdirectory of the directory in which you installed WorldUp.

The order in which nodes appear in the Scene Workview determines the order in which nodes are processed and the order in which graphical nodes are rendered. For each frame of the simulation, WorldUp begins processing the nodes in a top-to-bottom order (as displayed in the Scene Workview).

The scene graph enhances performance of the rendering stage (drawing the scene) because it facilitates spatial culling of the scene. In other words, WorldUp calculates which parts of the scene (or scene graph) are visible from the current viewpoint, and quickly rejects non-visible geometry before drawing begins.

The structure of your scene graph determines the parent/child relationships of the objects in the scene. For example, if you nest Node B beneath Node A, B becomes a child of A. A child always translates (moves) and rotates with its parent.

Observe the scene graph in the following figure.



In this example, Door, Wheel-1, Wheel-2, Wheel-3, and Wheel-4 are children of Car. If you move Car in the Simulation window, its children will move with it. Light-1, positioned directly beneath Root-1, will illuminate all the graphical objects in the scene since it is processed first. Spotlight-1, positioned below Car and its children, will illuminate Road, Building-A, and Building-B, but it will not illuminate Car and its children since they will have already been rendered before Spotlight-1 takes effect.

Notice, however, that because Spotlight-1 is not a child of Car, it will not move with Car. Attribute nodes only affect nodes that are at the same hierarchical level beneath it, or are nested within it. So, if Spotlight-1 was a child of Car, it would not illuminate the road or the buildings. To cause Spotlight-1 to move with Car, you would add behaviors. See Chapter 14, *The Behavior System*.

Working with Scene Graphs

This section describes the following:

- controlling the impact on a node's position and orientation when it is dragged in the Scene Graph pane.
- arranging nodes in the Scene Graph pane
- instancing nodes.
- removing nodes from the scene graph.

Controlling the Scene Graph Dragging Options

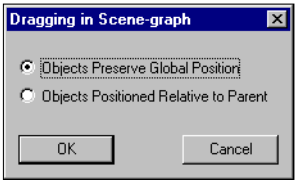
You can rearrange nodes in a scene graph by dragging and dropping them with your mouse. When you drag a node onto a new parent, you can control whether the node will maintain its global translation and rotation, or whether its translation and rotation will become relative to the parent node's reference frame.

For more information on reference frames, see "Coordinate Systems" on page 13.

To change the scene graph dragging options

1 Select Scene Graph Dragging from the Options menu.

The Dragging in Scene Graph dialog appears.



2 Select one of the scene graph Dragging Options buttons.

- *Objects Preserve Global Position* – The Translation and Rotation property values for the node object that is dragged are adjusted to allow the object to maintain its global position. The object will not change position in the Development window.
- *Objects Preserve Position Relative To Parent* – The Translation and Rotation property values for the node object that is dragged remain unchanged. Thus, if the parent object's reference frame is different than the global reference frame, the object that you drag in the Scene Graph pane will be repositioned in the Development window.

3 Click OK.

Rearranging Nodes

In Scene Workview, you can create an object by selecting a type in the Nodes pane and clicking on the Instantiate Selected Type button, or by dragging and dropping the type into the Scene Graph pane.

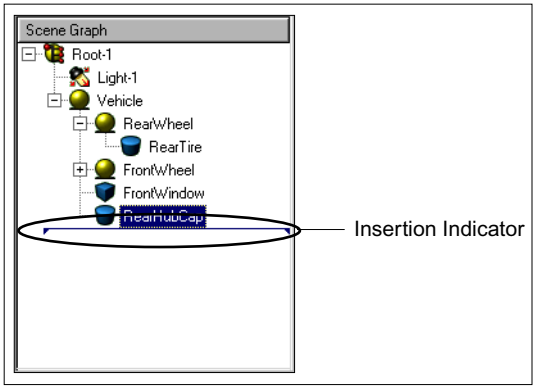
Using the drag-and-drop method, you can drop the object at the right place in the scene graph. When creating an object by clicking on the Instantiate Selected Type button, the newly created object is added at the end of the scene graph. You can then reposition the object in the scene graph later.

To reposition an object in the scene graph

1 Select the appropriate dragging option, as described in "Controlling the Scene Graph Dragging Options" on page 29.

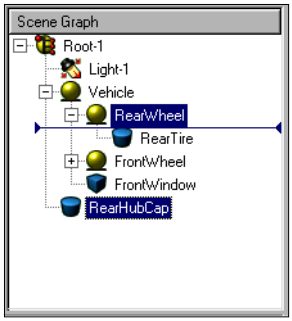
2 Click the object you want to reposition and drag.

The horizontal insertion indicator appears and moves with the object while dragging.



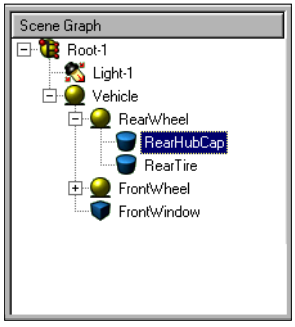
Note If you want to drage multiple nodes to the same location, hold down the CTRL key and select on each node that you want to drag.

3 While dragging, point the cursor over the node under which you want to place the dragged object.

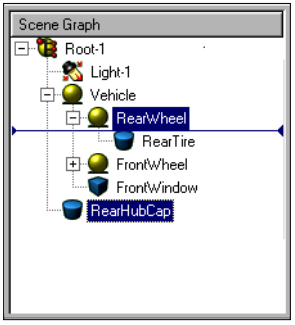


In the figure above, this is the RearWheel Group node. If a Group node already has some child node, its subtree expands as you bring the cursor over it.

4 Drop the object by releasing the mouse button while the insertion mark stays shortened as shown in above figure.



Note If you want to make the dragged object a sibling of the Group node, then drop the object while the insertion mark covers the whole pane which is its normal length as shown in the figure below. Moving the cursor to the right while the insertion mark is active shortens the length of insertion mark.



Holding down the CTRL key while you drag a node, duplicates the node. That is, a new object is created for each duplicated node and its corresponding node is positioned in the scene graph where you released the mouse.

Note Duplicating nodes is not the same as *instancing* nodes, as described in the next section.

Instancing Nodes

Instancing a node allows a node to appear multiple times in the Scene Graph pane and Development window without the performance impact of adding additional objects to your simulation. That is, each node instance references the same source object. Thus, if you manipulate one node, all instances of that node are manipulated in the same manner.

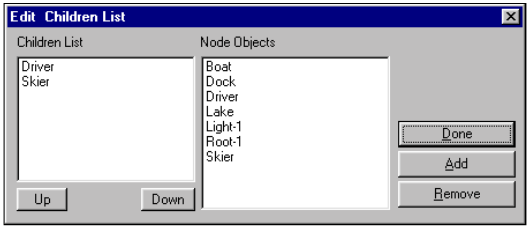
The only factor that differentiates node instances from one another is the Translation value of the *parent* for each instance. Creating instances of a single node under parent nodes with different translations is what causes the node to be rendered multiple times in various locations.

To instance a node

1 In the Scene Graph pane, click the node that you want to be the parent of the new instance you are creating.

2 In the Editable tab of the Property pane, double-click the Children property.

The Edit Children List dialog box displays.



The Children List box lists the nodes that already exist as children of the selected node. The Node Objects box lists all Node objects in the simulation.

Note It is possible to have Node objects in the Nodes pane with no corresponding nodes in the Scene Graph pane. See "Removing Nodes from the Scene Graph" on page 32.

3 In the Node Objects box, double-click the node that you want to instance.

The node moves to the Children List.

4 Click Done.

A new instance of the object appears in both the Scene Graph pane and the Development window, but no new objects are created in the Type pane. When you manipulate one node, all other instances of that node are manipulated in the same manner.

Note If the object from which you instanced the node is a graphical object and you cannot see it in the Development window, make sure that the parent nodes of each instance are positioned at different global locations. If they are not, the instances are being rendered in the same location.

Removing Nodes from the Scene Graph

Just as you can instance a node by making that node a child of multiple parents (see "Instancing Nodes" on page 31), you can also remove some or all instances of a node by removing the appropriate parent/child relationships.

When a node has no defined parents, that node no longer exists in the scene graph, and thus is no longer rendered anywhere in the Development window. However, the corresponding Node object remains in the Nodes pane.

To remove a node from the scene graph

1 In the Scene Graph pane, click the parent node of the node you want to remove.

2 In the Property pane, double-click the Children property.

The Edit Children List dialog box displays.

3 In the Children List box, click the node you want to remove.

4 Click Remove, then click Done.

That instance of the node is removed from the Scene Graph pane and is no longer rendered in the Development window. If multiple instances of the node exist and you do not want any instance of the node to be rendered in the Simulation window, repeat this procedure for each parent of each instance of the node.

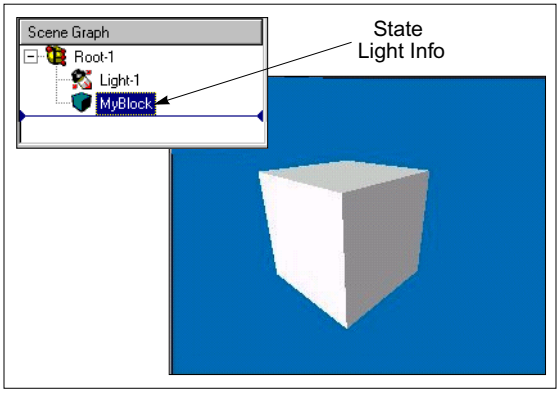
Note If you remove a node that has children, the children will also be removed from the scene graph.

To remove a Node object from the Nodes pane

➤ Manually delete the object using the Delete button.

How the Scene Graph is Traversed

The order in which nodes are processed is referred to as the *traversal order*. Since WorldUp uses a *state preserving* architecture, it is very important to note the way a scene graph is built in order to obtain the desired results when the simulation runs. In general, the scene graph is traversed from top to bottom and left to right. When an attribute node is hit, the contents of that node are placed onto the state (in programming terms a stack). When a graphical node is encountered during traversal, WorldUp looks at the state and applies what is there to the graphical node. When an organizational node is encountered, WorldUp looks at the contents of the node and decides how traversal should proceed based on the contents of the organizational node.



Scene Graph Traversal

For example, the figure above represents a simple scene graph constructed in WorldUp and shows the resulting simulation. Traversal begins at the Root node (Root-1) with an empty state, and proceeds to the Light node (Light-1). Upon encountering the Light node, WorldUp recognizes this as an attribute node and places its contents onto the state.

Traversal of the scene graph continues to the next node (MyBlock). At this point, WorldUp recognizes this node as a graphical node and takes what is currently on the state and applies it to the graphical node. In this way, the graphical node (MyBlock) is lit by the information contained in the Light node (Light-1) and rendered to the scene.

Traversal order continues back to the Root node, and the state is wiped clean, and WorldUp is now prepared for the next frame and a new traversal.

5

Working with a Project

In WorldUp, simulations are stored in project files that are saved as .UP files. When you first start WorldUp, a default project is created that contains the necessary components common to most simulations, including a universe, a window, a viewpoint, and a mouse. This chapter covers in detail the various file types and components that comprise a WorldUp project and how to manage them.

What is a WorldUp Project?

A simulation is comprised of many different components, such as models, textures, and sounds. A *project* is a collection of objects that represent these components. As such, there is typically a correspondence for example between a Geometry object in your simulation and the underlying geometry file that it represents. This applies to all of your simulation components except those which are built into WorldUp, such as sensors and primitive geometries (block, sphere, cone). These built-in objects need no external file to define them.

The following table lists the most commonly used WorldUp simulation objects and the file types they typically represent.

WorldUp Objects and File Types

Object/Component	Typical File Types
Imported Geometry	.nff, .3ds, .wrl, .jt, .flt
Imported Geometry Textures*	.tga, .jpg
Script	.ebs, .ebx
Sound	.wav
Behavior	.pup, .dll
Plugins (user defined types)	.dll
Dialogs (Script GUI components)	.dlg
Path	.pth

* Since imported geometry files typically contain their own references to texture files, these references are not contained in the .UP file.

Your project file preserves how these components are collected and assembled in your simulation and what state they are in. It does so through a database of Objects and Properties. See Chapter 9, *Objects and Properties* for more information.

When you save your project, it stores this database in an ASCII text file called an UP file (.UP). The UP file does not contain all of your simulation components, merely a description of their arrangement and file references. When you load your project, WorldUp reads this database of objects and property values, which tells WorldUp which additional files it needs to load and how to restore your simulation arrangement and state. It is very important that WorldUp is able to find all of the files referenced by objects listed in the UP file. Otherwise, the object cannot be re-created. Refer to "Configuring Directory Paths" on page 39 for more information on how to ensure WorldUp can locate all referenced files.

Creating, Loading, and Saving Projects

As mentioned above, WorldUp project files are ASCII text files describing the simulation contents, arrangement, and state. In addition to UP files, WorldUp provides a Simulation Packager that will package your UP file and all of your simulation's other files (models, textures, scripts, etc.) into a single, compressed file called a ZUP file (.ZUP), which is very useful when you want to redistribute your project to another computer. An uncompressed version of the ZUP file, the WUP file is also supported by WorldUp R5 for backwards compatibility with earlier versions of WorldUp. The following table describes the possible project file types.

WorldUp Project File Extensions

Project File Extension	Description
.UP	An ASCII text file that contains all of your simulation's objects and their properties.
.ZUP	A binary archive containing an .UP file as well as all other simulation files, including geometries scripts, textures, etc., as well as any user added files.
.WUP	Similar to a ZUP file, but not compressed. This file format is no longer used by WorldUp, but supported for backward compatibility.

Creating a New Project

To create a new project

➤ Click the New Project button.

A new, empty project appears in the Project Workview.

Loading a Project

You can load UP, ZUP, or WUP files for editing.

To open a project file

1 Click the Open Project button in the main toolbar.

A File Open dialog box appears.

2 In the Files Of Type drop-down box at the bottom, confirm the `Simulations (*.Up, *.ZUP, *.WUP)` file filter is selected.

3 Navigate to the appropriate drive and directory and double click the UP, ZUP, or WUP file you want to open.

Alternatively, if the project that you want to open was one of the last eight files opened in WorldUp, its name appears in the list at the bottom of the File menu. Select the file name to open it. You can also open any project by dragging the project file into WorldUp from any file browser, such as Windows Explorer.

If you open a file of type ZUP or WUP, WorldUp first unpacks all of the files contained in the archive into a directory with the same name as the project in the WuCache directory under your windows directory. After unpacking these files, WorldUp then proceeds to load the UP file from that directory. Since this project is now unpacked in the WuCache directory, the next time you load a ZUP or WUP file with the same name, it will first look in that directory. If the project already exists there, WorldUp loads the existing one instead of the ZUP file you selected. You should be careful to delete the file in the WuCache directory if this is not the file you want opened.

Saving a Project

To save your Project file

➤ Click the Save button.

Exporting a Project

In addition to saving your simulation, you can choose to export your simulation as a compressed, single archive that contains all project related files (a ZUP file).

To export your file

➤ Select Export Project As>WorldUp Project from the File menu.

For more information about exporting your project as a ZUP file, refer to Chapter 20, *Publishing Your Application*.

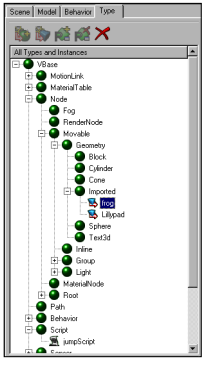
Importing An Existing WorldUp Project

In WorldUp you can import objects and all elements associated with those objects from one project into another. You import an existing project using the Resource Browser.

Resource Browser

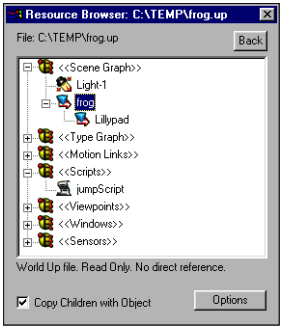
When you open an UP file in the Resource Browser, all elements of that project, such as scene graph hierarchies, object types, and attached scripts, display in a Resource Browser dialog box. When you drag any project element from the Resource Browser into the Project Workview, that element retains and copies with it any other elements to which it is related.

For example, let's say two graphical objects are in a project (FROG.UP), which are called Frog and Lillypad.



FROG.UP as it appears in the Type Workview

In the project's scene graph, Lillypad is a child of Frog. The Frog object may also have a script attached to it which is called JumpScript.



FROG.UP as it appears in the Resource Browser

In another project, if you load FROG.UP into the Resource Browser and drag only the Frog node into the Scene Graph pane of the Scene Workview, the current project inherits not only the Frog object, but also the Lillypad object and the JumpScript object. The Lillypad is automatically positioned as Frog's child in the scene graph, and the JumpScript object is automatically added to Frog's Task list.

To import elements from a UP file:

1 On the main toolbar, click the Import Project button.

The main view of the Resource Browser appears.

2 Click Add Resource.

The Open dialog box appears.

3 Navigate to the directory containing the UP file that you want to import.

4 Double-click the file name.

The elements of the imported project display in a new view of the Resource Browser, categorized under the roots in the following table.

Resource Root	Contains
Scene Graph	The hierarchical structure of all nodes in the .UP file
Type Graph	All object types and the inheritance structure in which they were created
Motion Links	All MotionLink objects in the .UP file
Scripts	All Script objects in the .UP file
Viewpoints	All Viewpoint objects in the .UP file
Windows	All Window objects in the .UP file
Sensors	All Sensor objects in the .UP file

Resource Options for UP files

When you open an UP file in the Resource Browser, you can access the resource options by clicking the Options button at the bottom of the Resource Browser dialog box. The following table shows the options that are available.

Resource Option	Description
Reload Resource	WorldUp reloads the .UP file into the Resource Browser. Suppose you are running multiple sessions of WorldUp. In one session you have opened FROG.UP. In the other session, the Resource Browser contains a resource for FROG.UP. As you modify FROG.UP in the first session, you can use the Reload Resource command in the second session to update any changes that have been made.
Remove Resource from Project	Removes the .UP file from the Resource Browser. All unused resources are automatically removed when you close the project. When you drag elements of an .UP file from the Resource Browser into your current project, they are actually copied into the current project, instead of the elements referencing the UP resource.

Configuring Directory Paths

It is crucial that your project be able to find all files referenced by objects in your simulation. Some problems that may occur as a result of incomplete path settings are:

- Models load without their textures
- Models fail to load
- Scripts fail to load
- Sounds fail to load

Typically, WorldUp will displays a message in the Status window while loading a project if it cannot find a file that was referenced either by an object in the UP file or by a model. Plug-ins and other DLLs, however, may not always be so obvious.

In order for WorldUp to find the referenced files, they must either exist in the same directory as the UP file (one advantage to packaging your simulation as a ZUP file since all of its contents are extracted into a single directory), or the path names in which they are located must exist in the Directory Paths To Search list. Each search path can be stored as a Project Path or a System Path.

System Paths vs. Project Paths

A *System Path* is a search path that is available for any WorldUp project. That is, the path is added to the Directory Paths To Search list for the entire WorldUp system which is preserved in the system registry.

A *Project Path*, on the other hand, is a search path that is available only for a certain project (UP file). That is, the path is added to the Directory Paths To Search list for the given UP file only.

Depending on the location of the path, a Project Path may be stored either as an *absolute* path or a relative path. If the directory you want to add as a Project Path is nested within the directory in which the opened UP file is located, the path is stored as a relative path. For example, if your UP file is located in C:\DRIVESIM and you are adding the C:\DRIVESIM\MODELS\TRUCKS path as a Project Path, it is stored as:

```
models\trucks
```

If you convert that path to a System Path, it is stored as the absolute path:

```
c:\drivesim\models\trucks
```

If you plan on redistributing your simulation onto other machines, it is very important to store your search paths with the project, and that these paths are relative, since you do not necessarily know where on the target machine your simulation will be installed.

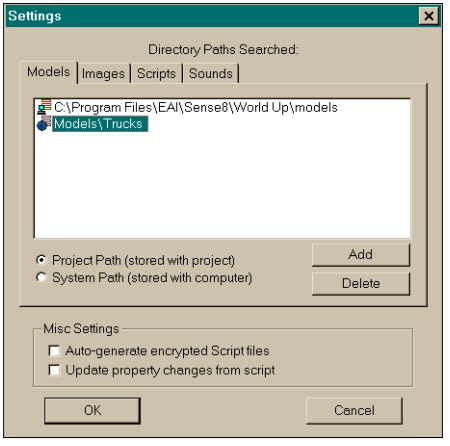
By storing nested Project Paths as relative paths, you are given the freedom of moving your project directories to another location without having to set new search paths. In the relative path example above, if you moved the DRIVESIM directory and its subdirectories to the D drive, your project would still be able to locate the files in the MODELS\TRUCKS subdirectories since the path is always relative to the UP file's directory.

Note If the directory that you are adding as a Project Path is not nested within the UP file's directory, it is stored as an absolute path.

To add a directory path to the search list

1 Select File Access Settings from the Options menu.

The Settings dialog box appears.



2 Click the appropriate tab for the type of path (such as Models) that you want to add.

3 Click Add.

The Open dialog box displays.

4 Select the directory path you want to add and click OK.

A default storage mode is assigned to the path and is reflected by both the selected radio button (System Path or Project Path) and the icon preceding the path

5 To change the storage mode, click the appropriate radio button.

Note You can change the storage mode for any path at any time.

To delete a directory path from the search list

1 Select File Access Settings from the Options menu.

2 In the Settings dialog box, click the appropriate tab for the type of path (such as Models) that you want to delete.

3 Click the path you want to delete and click the Delete button.

4 Click OK.

Global Simulation Settings

Certain settings can be considered global to your simulation, which is to say they affect more than just a single object. These settings typically affect the way things are done, how things are drawn, etc. These global settings affect your entire simulation, such as ambient light, texture mip-mapping styles, sound device used, etc. Since these global simulation settings must persist within your project, they are all stored in the project's single Universe object. To help you understand and configure your global simulation settings, the remainder of this section covers the Universe object's properties.

Rendering Options

Rendering is the process of calculating and then drawing images on a screen. WorldUp provides several options related to rendering.

Turning Rendering On or Off

By default, rendering is turned on in WorldUp.

To toggle rendering on and off,

➤ Click the Rendering On/Off  button.

When you turn rendering off, the current image in the Development window is no longer updated. Thus, all navigational methods are disabled until you turn rendering back on.

If you are developing a simulation that is graphics-intensive, you may find it useful to turn off rendering at times when it is not important to see rendering updates made in the Development window. This will free up resources to improve performance as you work in the Project Workview, or as you work in another application while WorldUp is still running.

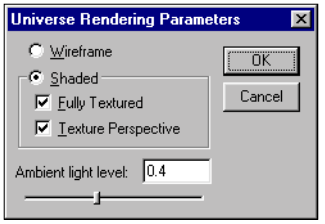
Setting Rendering Parameters

You can change the way in which the simulation is rendered by modifying the rendering parameters. For example, you could render all geometries in the simulation without textures to increase the frame rate as you work.

To set rendering parameters

1 Click the Set Rendering Parameters  button.

The Rendering Parameters dialog box displays.



2 Set the desired options (described below) and click OK.

Option	Description
Wireframe	Renders geometries as points and lines, without texturing and shading. This option allows for the fastest rendering possible.
Shaded	Shades each polygon of a geometry according to the intensity of light striking the polygon and the Material property value applied to the geometry.
Fully Textured	Renders geometries with their textures fully shaded according to the intensity of the light. This option is only available when Shaded is turned on.
Texture Perspective	Shows geometries with textures and scales the textures when the viewpoint changes. This option is only available when Shaded is turned on. **Note:** For some systems, textures are always scaled to the viewpoint, regardless of this WorldUp option.
Ambient Light Level	A number between 0.0 and 1.0 that adjusts the brightness of ambient light, with 0.0 being total darkness and 1.0 being total brightness.

Note The simulation can be running or idle when you change the rendering parameters.

Displaying Rendering Performance

By default, the status bar at the bottom of the WorldUp window displays the current frame rate for the simulation when rendering is turned on. This is the number of times per second that the screen image is redrawn.

To hide or show the rendering performance

➤ On the View menu, check or uncheck Show Frame Rate.

Note A number of factors can affect the frame rate of your simulation. Chapter 19, *Tips and Tricks* explores some of these factors and provides suggestions on how to improve your frame rate.

The Universe Object

Every WorldUp project has a single Universe object. This object contains all of the properties that effect your entire simulation.

The following table lists the properties associated with the Universe object.

Universe Object Properties

Property	Description	Valid Values
Ambient light	Sets the amount of ambient light in the simulation. Ambient light is light that effects all polygons equally.	0.0 to 1.0 – A setting of 0.0 effectively turns the ambient light off. 1.0 is full intensity.
Anti-aliasing	Turns on off anti-aliasing. This only works for specifically supported video hardware.	True or False
Audio Device	Sets the .dll used to communicate with your sound card. By setting it to None you will release the sound card so that other programs may access it.	None – No audio device in this simulation. Default – Use windows default sound device (winmm). Winmm allows you to play only one sound at a time. DiamondWare – Use Diamond Ware sound device. This allows you to play four sounds at one time. CRE – Use Crystal River Engineering sound server.
Audio Listener	Sets the position from which a sound is heard.	Any viewpoint in the simulation
Audio Rolloff	Sets the distance at which sound becomes inaudible. As this value increases, sounds in the universe become louder.	Any positive decimal value
FrameRate	Sets the number of frames per second (fps) displayed during a running simulation.	This is calculated by WorldUp and is read only.
Mipmap Mag Filter	Sets the the magnification filter style for texture mapping.	Nearest Linear
Mipmap Min Filter	Sets the minification filter stye for texture mapping.	Nearest Linear NearestMipMapNearest LinearMipMapNearest NearestMipMapLinear LinearMipMapLinear
NoAutoAlpha	Disables automatic use of black pixels as the alpha channel.	True – Don't set black as transparent False – Set black to be transparent
Perspective Texture	Enables or disables perspective-correct texturing.	True – Perspective-correct texturing enabled False – Perspective-correct texturing disabled
Sensitivity Percentage	Sets the mouse sensitivity as a % of the universe diameter.	Any non-negative value

Property	Description	Valid Values
Shutdown script	Sets the filename of a script that you want to run automatically when you stop a simulation.	Any valid script. The script must contain a main subroutine.
Startup script	Sets the filename of a script that you want to run automatically when you load a project (.up) file.	Any valid script. The script must contain a main subroutine. This subroutine will be run once when the project is loaded.
Tasks	Name(s) of the script(s) to be executed once per frame during the simulation.	Any valid script. The script must contain a task subroutine. The task will be run once each frame of the simulation.
Texture Memory Used	Sets the amount of texture memory the textures are using.	N/A
Textured	Enables or disables texture mapping in all views.	True – Texture mapping enabled False – Texture Mapping disabled (textures are not drawn)
User script	Sets the name of the script to run whenever you choose User Defined Script from the Simulation menu or click the User Defined Action button on the toolbar.	Any valid script with a main entry point
Wireframe	Indicates whether the universe is rendered with wireframe parameters. When wireframe rendering is True, WorldUp displays objects as points and lines, removing all texturing and shading.	True – Wireframe enabled False – Wireframe disabled

6

A Quick Tour

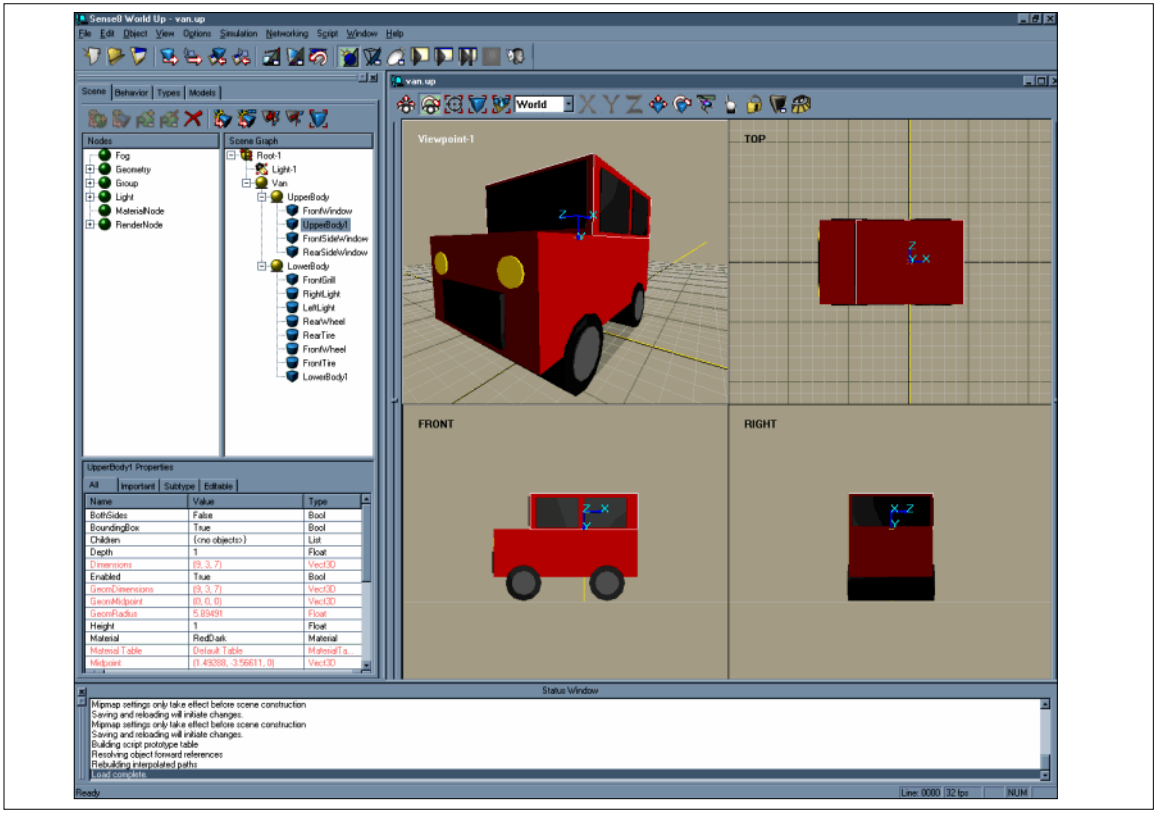
This chapter contains four guided tours that are intended to quickly familiarize you with many of the procedures involved in creating a WorldUp simulation.

Completed versions of each guided tour are located in the Tutorials subdirectory of the directory in which you installed WorldUp. You can use these simulations as a reference, or as a starting point should you decide to skip the previous tutorial.

Tutorial 1: Creating a Model

In order to familiarize yourself with the WorldUp development environment, we start by creating a vehicle from basic primitives, such as a block and cylinder. The figure below shows the completed vehicle.

This tutorial will show you how to set up the development windows, navigate the scene, and position and manipulate objects using the new user interface in WorldUp Release 5.



Completed Vehicle in the WorldUp Window

Getting Started

To begin

1 Create a new directory in where you will save your WorldUp simulations as you work.

2 Start WorldUp, or click File, New Universe if WorldUp is already running.

The window that is displayed when you start a WorldUp session is known as the WorldUp Development window.

3 If desired, enlarge the Development window.

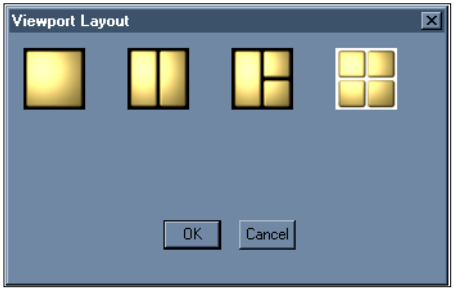
Setting up the working environment

First let us arrange the working environment. In WorldUp multiple viewports can be set up in development window to view a scene from various angles. Multiple viewports facilitates you to position objects in 3D space correctly and accurately.

To add more viewports

1 Click on Display Options button on the navigation toolbar of the Development window.

2 Click Configure Viewports.



3 Select four viewports and click OK.

4 Click OK to close the Display Properties dialog.

You have now added four viewports to the Development window, showing views from the default viewpoint (viewpoint-1), top, front, and right. The default viewpoint shows the perspective view and the others orthographic views. At any time only one viewport is active and this is the viewport on which navigation operations are applied. The active viewport is highlighted with a white border.

Creating the body

We begin by creating the lower body of the vehicle using a block. The WorldUp development environment consists of main menu and main toolbar on top, project window on left, development window on right and status window at the bottom. The project window consists of four various Workviews; Scene Workview, Behavior Workview, Type Workview and Model Workview. Workviews are logically organized folders for doing various tasks.

We'll be focusing on Scene Workview in this chapter. The Scene Workview contains the Nodes pane showing types and instances on the left, the Scene Graph pane showing the scene graph on the right, and the Property pane on the bottom showing properties of the selected node.

First, we must create a Vehicle group node where we will put all the vehicles parts.

To create the Vehicle group

1 Select the Scene Workview tab, if not already selected.

2 Select Group in the Nodes pane.

3 Drag and drop the Group node into the Scene Graph as a child of Root just below Light-1.

Note Light-1 must be the first child of the Root node before all other nodes because lighting is applied only on nodes below the Light-1 node.

4 Select anywhere on the Name row in the Property pane.

5 Select the name Group-1.

You will see an edit box appear.

6 Change Group- 1 to Vehicle.

Next, we create the lower body of the vehicle.

To create the lower body

1 Expand the Geometry node in the Nodes pane to show its sub-types.

2 Select the Block sub-type.

3 Drag and drop the Block to the Scene Graph as a child of the Vehicle group node.

This creates (instantiates) a node named Block-1 under Vehicle.

4 Select anywhere on the Name row in the Property pane.

5 Select the name Block-1.

You will see an edit box appear.

6 Change Block-1 to LowerBody.

7 Click on the Material property and select Red material.

8 Edit the Stretch property to be (12, 4, 7).

9 Edit the Translation property to be (0, -3.5, 0)

This moves the block above ground level.

Now let's create the upper body of the vehicle.

To create the upper body

1 Create a new Block as we did for the lower body.

2 Name the new Block UpperBody.

3 Apply Red-Dark material.

4 Set the Stretch property to (9, 3, 7).

To look at the vehicle closely in all views

1 Click in a viewport to make it active.

2 Click the Zoom All button.

3 Repeat this for each viewport.

Now we need to place the upper body above the lower body.

To move the upper body

1 Select UpperBody in the Scene Graph.

2 Change the Translation property of UpperBody to (1.5, -7, 0).

You can also use one of the following methods to move the upper body:

- Double click on Translation property and a Translation tool pops up. Use slider bars to set the X value to 1.5 and Y value to –7.
- Click the Translate Object button. While UpperBody is selected in the Scene Graph, click the Lock Selected button. Then click and drag the UpperBody to translate it. To move in Y direction, use right mouse button.

Note Once the Lock Selected button is pressed, the object currently selected remains selected even if you click on an empty space in Development window.

To learn more about the Translate Object and other tools in the Development window toolbar, see Chapter 8, Development Window – Navigation and Manipulation.

Saving the Model

You can save the vehicle you have created as a project file (.UP) or export it to VRML.

To save your model as a project

➤ Save As from the File menu.

In this example, let us save the vehicle you have created so far as VRML1.

To export your model as VRML

1 Select Export Project As > VRML 1.0 from the File menu.

2 Name the model `vehicle.wrl` in the project directory you created at the beginning of this tutorial.

It is a good practice to save your work frequently to avoid data loss or to revert back to a previously saved model.

Working on the Wheels

Now that the body of the vehicle is in place, let's create the tires and wheels for the vehicle using cylinders.

To create the front tire

1 Select the Cylinder sub-type from the Nodes pane.

2 Drag and drop the Cylinder to the Scene Graph as a child of the Vehicle group node.

3 Name the Cylinder FrontTire.

4 Change the Material property to MatteBlack.

5 Set the Radius property to 1.5.

6 Set the Height property to 7.2.

7 Rotate the FrontTire on the X axis using one of the following two methods:

- Double click on the Rotation property of FrontTire to get the rotation tool. Slide the Pitch (X) slider bar to right to set it to 90 degrees. Click on "Snap to Every [15] degrees" to rotate faster and in steps of 15 degrees. You can change this to 30, 45 or 90 degrees if you want to reduce the number of steps to reach 90 degrees.
- Select the Object Rotate button, press X on the toolbar to set the axis of rotation as X-axis. Press Lock Selected to keep the selection intact. Click on the cylinder and move the mouse up or down to rotate the object until it is placed horizontally.

Now creating the wheels and rear tire are easy. Right click on FrontTire in the Scene Graph and select duplicate. Name this new cylinder FrontHubCap, set its Initial Radius to 1.0, height to 7.3 and change its material to Gray.

8 Save your work as `vehicle.wrl` in VRML1.0 by selecting File > Export Project As > VRML1.0.

Grouping the Objects

Grouping of objects and assigning parent-child relationships between objects are essential in modeling and creating simulations. For example, moving one node underneath another node creates a parent-child relationship between those two objects. When you move the parent object, the child object moves with the parent.

For more details about Groups, see Chapter 4, Organizing Your Scene.

In our model, let's group the wheel and tire as a single group node.

To group tire and wheel

1 Select Group in the Nodes pane.

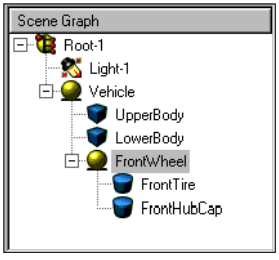
2 Drag and drop Group into the Scene Graph below Vehicle.

3 Name this group node FrontWheel

4 Drag and drop FrontTire under the FrontWheel group node to make it a child of FrontWheel.

5 Drag and drop FrontHubCap to FrontWheel in the same manner as step 4.

Your scene graph should look similar to the following figure.



You can also select both FrontTire and FrontHubCap at the same time by holding down the shift key when selecting and drag both to FrontWheel in one step.

Now, let's move FrontWheel to its correct position.

To move FrontWheel

1 Select FrontWheel in the Scene Graph.

2 Set the Translation property to (-2, -1.5, 0).

To move front wheel using the tools

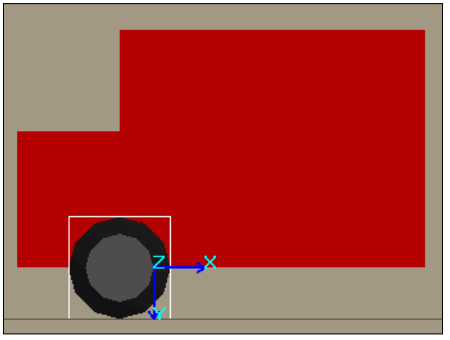
1 Click the Lock Selected button while FrontWheel is selected.

2 Select the Translate Object button in toolbar.

3 Click on Constrain X to move it only in X direction.

4 Using the left mouse button, move FrontWheel to the left.

5 Use right mouse button to move it Up.



Now that you have successfully created the front wheel, let's work on the rear wheel.

To create the rear wheel

1 Create a Group node as child of Vehicle.

2 Name the new Group node RearWheel.

3 Right-click on FrontTire and select the Duplicate button.

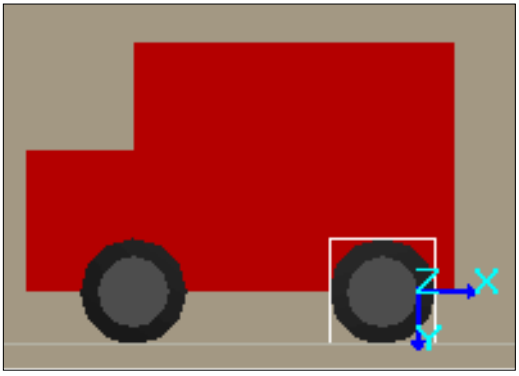
4 Name the copy RearTire.

5 Drag and drop RearTire into the RearWheel group node we just created.

6 Duplicate FrontHubCap, and name it RearHubCap.

7 Group RearHubCap with RearTire under RearWheel.

8 Just as we moved FrontWheel to its correct location, move RearWheel to its final position using the Object Translate tool or by setting the Translation property to (5, -1.5, 0).



Note By duplicating and resizing FrontWheel and RearWheel parts, an advanced user can create left and right wheels for each.

9 Save your work as `Vehicle.wrl`.

Creating Windows

Next, let's add windows to the vehicle.

To create the Front Side window

1 Add a new Block to the Scene Graph and name it FrontSideWindow.

2 Set the Material property to MatteBlack.

3 Set the Stretch property to (4, 2.5, 7.1).

4 Translate the window to (-0.75, -7, 0).

Now we can duplicate this object to create the rear side window and the front-rear window.

To create the rear side window

1 Duplicate FrontSideWindow.

2 Name it RearSideWindow.

3 Translate RearSideWindow to (3.6, -7, 0.)

To create the front-rear window

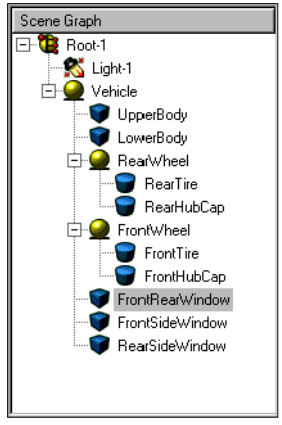
1 Duplicate FrontSideWindow.

2 Name it FrontRearWindow.

3 Set the Stretch property to (9.1, 2.5, 6.5).

4 Translate the window to (1.5, -7, 0).

Your scene graph should look similar to the following figure.



Creating Lights

You can create front lights using cylinders. Set a relatively smaller Z value compared to X in the Stretch property of the light to get an elliptical shape as shown in fig.

To create the left light

1 Add a Cylinder to Vehicle and name it LeftLight.

2 Set the Stretch to (0.5, 0.5, 0.6)

3 Rotate it 90 degrees in Z.

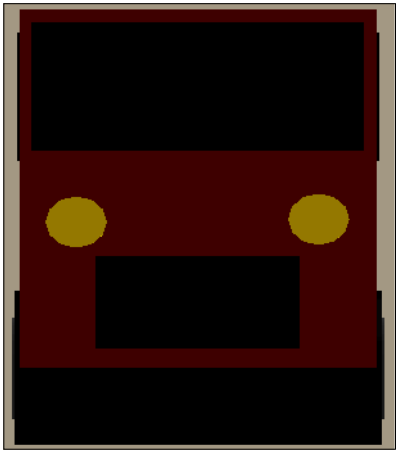
4 Set the Material to Gold.

5 Translate the LeftLight to (-6, -4.5, -2.5).

To create the right light

1 Duplicate LeftLight and name it RightLight.

2 Translate it to (-6, -4.5, 2.5)



To create the front grill

1 Add a new block to the Scene Graph and name it FrontGrill.

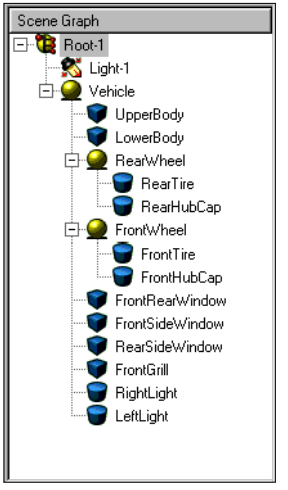
2 Set the Stretch property to (1, 1.8, 4).

3 Set the Material to MatteBlack.

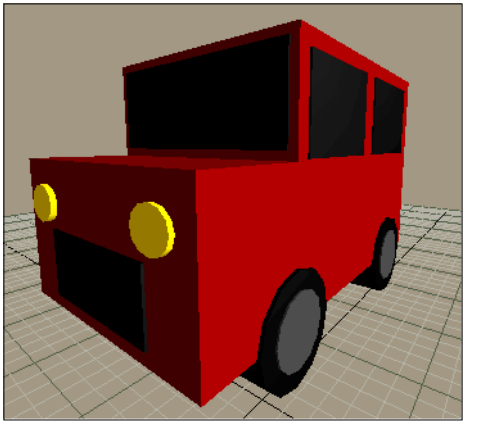
4 Translate it to (-5.6, -3, 0).

5 Save your finished work as vehicle.wrl in VRML 1.0 by clicking on Export Project As> VRML1.0 from the File menu.

The final Scene Graph should look similar to the following figure, though the nodes may not necessarily be in the same order.



The completed model should look similar to the following figure.



Congratulations! You have successfully finished creating your first model in WorldUp. In this Tutorial, you have learned to:

- Use the Scene Workview to add objects to the Scene Graph.

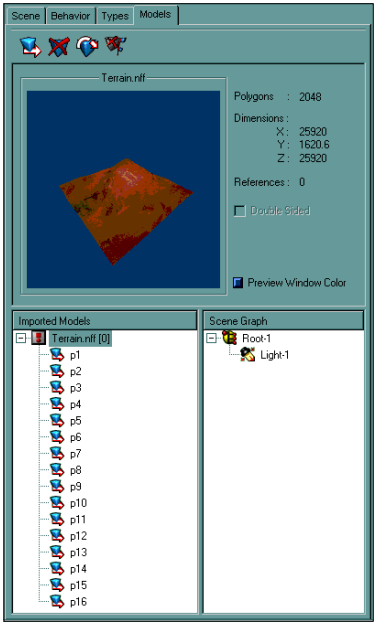
- Set up multiple viewports in the Development window.
- Create models using primitives such as Block and Cylinder.
- Use navigation and manipulation tools to view, translate, and rotate objects.
- Use the Property pane to edit an object's properties.
- Group nodes and assign parent-child relationships between objects.
- Save the model.

Tutorial 2: Importing a Model

In the previous Tutorial, we learned how to create a model using basic primitives provided with World Up. To create a detailed simulation you will need to bring in already available models to the WorldUp simulation environment. In this tutorial, we will import some models, including the vehicle you created in last lesson, to build a driving simulation over a terrain.

Using Model Workview

By now you are familiar with the Scene Workview, which displays the hierarchical structure of your scene and its properties. It allows you to create new objects from available types. The Model Workview allows you to import and preview geometric models and add them to your scene.



The Model Workview

For more details on the Model Workview, see Chapter 7, Using the Workviews.

Importing Terrain

First, let's import something to stand on. Before importing a model, we create a Group node called Terrain to group the related geometries together.

To create the Group node

1 Select the Scene Workview tab.

2 Drag and drop a Group node below Light-1 and name it Terrain.

As we are going to create a high polygon scene let us turn off shadows and grids.

To turn off shadows and grids

1 Click on Display Options button.

2 Check the boxes next to Grid and Shadows to toggle them off.

3 Click on Viewport Configuration and select single viewport.

4 Click OK.

5 Click OK on Display Options dialog box to save the changes.

To import the terrain model using Model Workview

1 Click on the Models tab to select the Model Workview.

2 Click on Import New Model button and select terrain.nff from the Tutorials\Models directory.

3 Click OK

4 Click on Add New Path to the Search Path.

5 Click OK in the Import Model Parameters dialog box.

This will show the terrain model in the Preview Window and its hierarchy in the Imported Models pane. Other details about the model are displayed next to the Preview Window.

6 Select Terrain.nff [0] in the Imported Models pane.

7 Drag and drop it to the Scene Graph pane as a child of the Terrain group node we just created.

This adds terrain model to the scene graph and the model shows up in the Development window.

8 Click on Zoom All button.

The terrain disappears from the development window. This is because the window clips the terrain. In order to fix this we need to set the far (yon) clipping plane of the window to a high value.

To set the clipping plane

1 In the Project window, click on Type tab to select Type Workview.

2 Under Window > DevWindow, select DevWindow-1.

Make sure you expand the VBase node, if it's not already expanded.

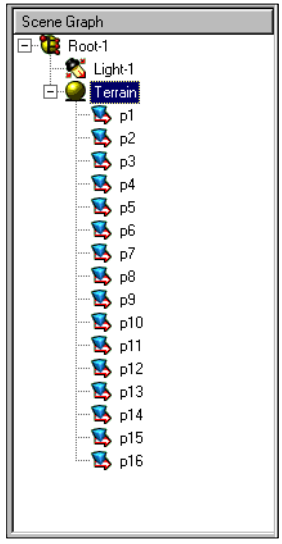
3 Look at its properties in the Property pane and set the Yon Clipping to 50000.

You should see the terrain appear in Development Window. Now we have expanded the DevWindow view volume.

4 Repeat this for the Application Window also by selecting Window > Window-1 and setting its Yon Clipping value to 50000.

5 Save your project as Vehicle.up by selecting Save As from the File menu.

You have successfully imported terrain into your simulation. The following figure show the current state of the scene graph.



Importing Walls

Next, we will build walls around the terrain.

To build the walls

1 Create a Group node in the Scene Workview as you did before, and name it Walls.

2 From the Model Workview, click the Import New Model button.

3 Select wall.nff from the Tutorial directory and click OK.

4 Click OK on the Texture warning as well.

5 Drag and Drop wall.nff from Imported Models pane to Scene Graph as child of Walls.

6 Repeat Step 3 through 5 three more times to add a total of four walls to the scene.

7 Go to Scene Workview and rename Block-1, Block-2, Block-3 and Block-4 under Walls as Wall-1, Wall-2, Wall-3 and Wall-4 respectively.

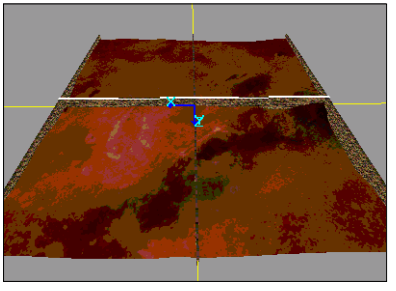
To place the walls correctly

1 Select Wall-1 and set its translation property to (13000, 0, 0).

Or, use a manipulation tool to translate it. To do this click on Object Translate button and constrain translation to left and right by clicking on X. Lock Wall-1 selection by clicking on the Lock Selected button. Drag mouse to the left to place the wall at the left end of terrain.

2 Select Wall-2 and set its translation to (-13000, 0, 0) Or use Object Translate to move it to the other end of the terrain.

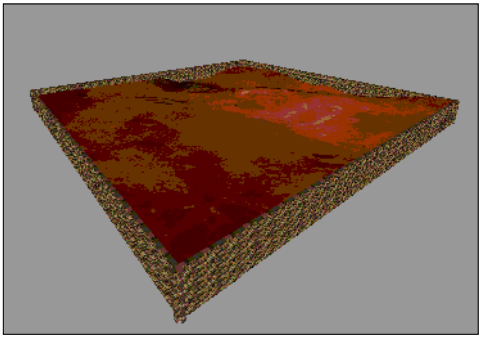
3 Select Wall-3 and rotate it 90 degrees in Y axis using a method explained in previous section.



4 Apply a translation of (0, 0, 13000).

5 Select Wall-4 and rotate it 90 degrees in Y-axis

6 Set Wall-4's translation value to (0, 0, -13000).



7 Select Save from the File menu to save your Project.

You have now put a border around your territory.

Importing Vehicle

The last step is to bring in the vehicle created in the pervious tutorial.

To import the vehicle

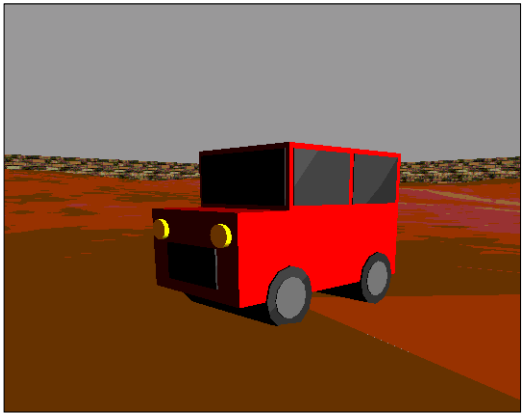
1 Using the Model Workview, import Vehicle.wrl from the \Tutorials directory.

2 Add it to the Scene Graph.

3 Switch to the Scene Workview and while keeping Vehicle node selected, click the Lock Selected button.

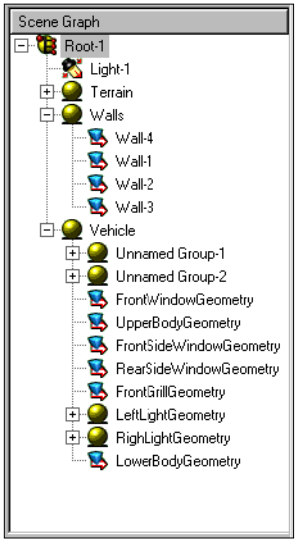
4 Click the Zoom to Selected button to zoom to vehicle.

5 Translate it up to around (0, -112, 0) to bring the vehicle above the terrain.



6 Select Save from the File menu to save your project.

You have now completed the scene building for your simulation. The scene graph should look similar to the following figure.



In next tutorial, you will add behaviors to your simulation.

Summary

In this tutorial you have learned how to:

- Use the Model Workview to import models
- Add imported models to your scene
- Work with large databases
- Adjust the Window/Viewport clipping plane to fit the whole scene
- Save your project

Tutorial 3: Using Behaviors

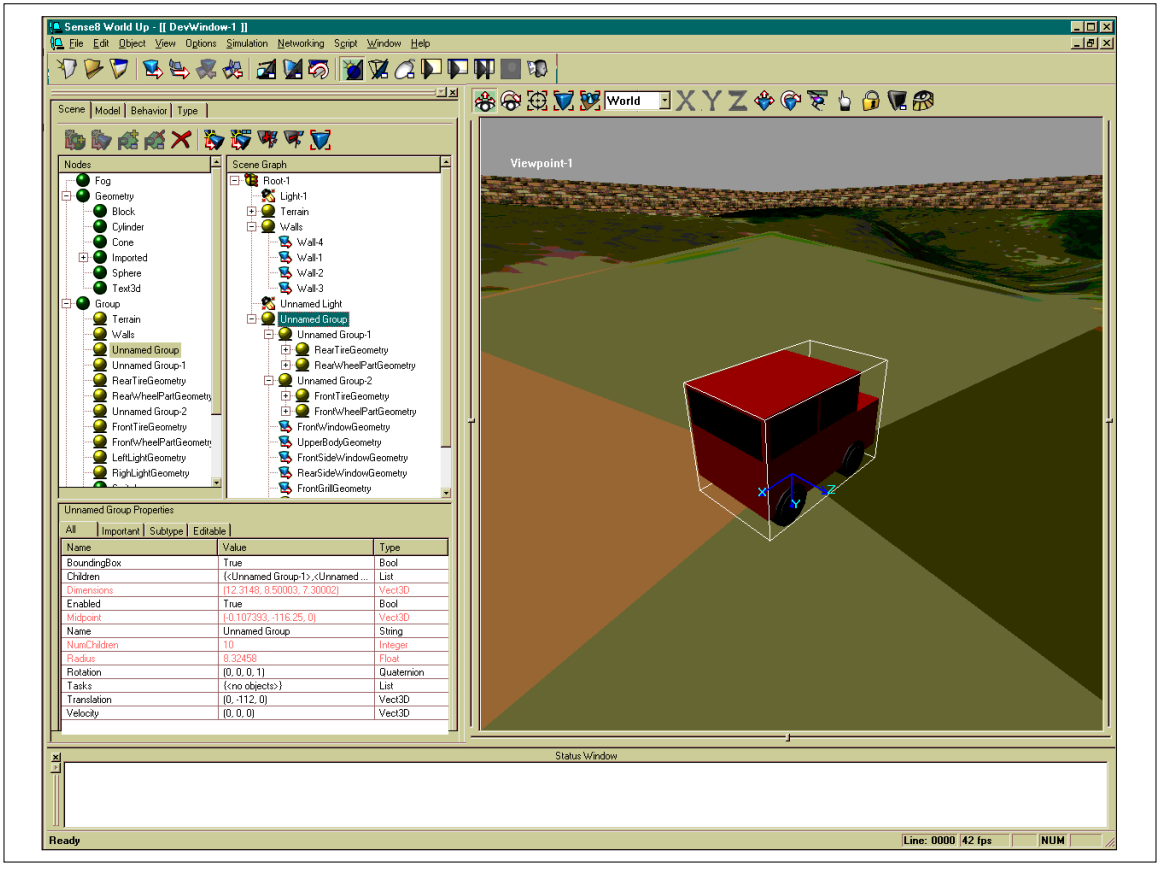
In this tutorial we will bring our simulation to life through the application of behaviors. In WorldUp, a behavior is generally defined as some activity that is applied to or demonstrated by any WorldUp Object within the Simulation. In our evolving tutorial simulation, we're going to add some classic simulation activities, including interactivity using a Sensor behavior, collision detection, and terrain following.

To begin

➤ Open your WorldUp project file from the previous tutorial.

You can also use the template file for this tutorial, `Tutorial2_final.up` in the `\Tutorial` directory. Either project should get you started where Tutorial 2 left off.

The project should look something like the following figure.

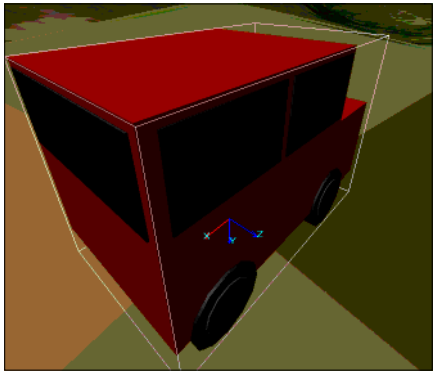


Tutorial2_final.up in the WorldUp Window

Lesson 1: Preparing your Geometry

Behaviors involve moving objects. When moving objects, we typically think in directions (such as forward, backward, and right) and orientations (yaw, pitch and roll). 3D graphics conventions typically map yaw as a rotation about the Y-axis, pitch as a rotation about the X-axis, and roll as rotation about the Z-axis. This convention thus defines "forward" (the direction the "nose" of the vehicle or viewpoint points) as the + Z-axis and backward as the –Z-axis.

In our project, we can see that our van's idea of a *forward* direction, however, is down it's –X-axis:



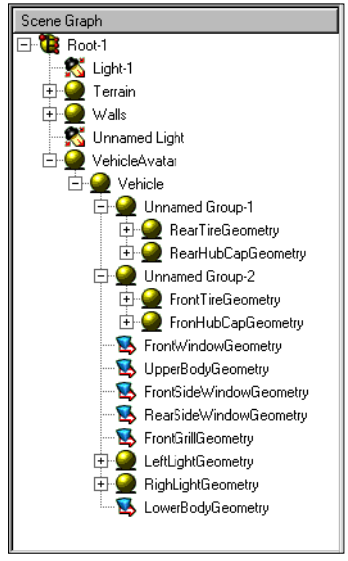
This is a typical situation, especially when importing geometries for which you had no control over how they were modeled. This is not, however, a problem. This short lesson will show you how to easily re-orient your model without modifying it at all.

To orient your model

1 Create a new Group node and name it VehicleAvatar.

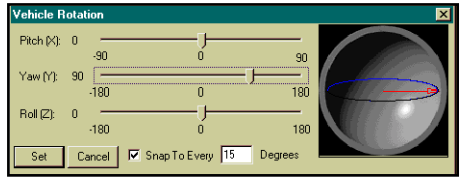
2 Make your Vehicle group a child of VehicleAvatar by dragging your Vehicle group node onto the VehicleAvatar Group node.

Your Scene Graph should now look something like the following figure.



3 Select your Vehicle Group and rotate +90 degrees about it's Y-axis.

You can do this either interactively in the render window with the Rotation controls or by double clicking it's rotation property and using the Rotation dialog.



4 From the Development window toolbar, select Local from the coordinate frame drop down box.

Now take a moment to examine the axes drawn in the render window, and their difference when the Vehicle is selected versus when the Vehicle Avatar is selected.

The Vehicle group shows the Van still pointing down it's –X axis. The VehicleAvatar group, however, shows the van pointing down the VehicleAvatar's +Z axis. This is what we'll need in order to move it in the direction we consider "forward."

In this lesson, we have learned how to adjust an object's orientation relative to another object's orientation. In essence, we are using the group node we created as a sort of dummy helper that allows us to change the van's orientation "with respect to" another object (the VehicleAvatar group). In the next lesson, we'll see why this becomes useful.

Lesson 2: Adding a Sensor Behavior

Sensors, such as Joysticks, Mice, and Trackers, are the window of interactivity between the user and the simulation. WorldUp supports a vast array of Sensors. For this lesson, we will focus on using a Behavior that allows the user to control the van with the Mouse.

To add a behavior

1 Select the Behavior Workview tab in the Project Workview.

This is where you will be doing most if not all of your behavior related work.

2 In the Behaviors pane, expand the PluginActions and PluginTriggers types to examine the standard set of behaviors that come with WorldUp.

3 Select the MouseDriver Action type and examine it's properties in the Property pane.

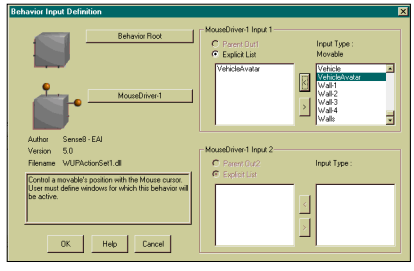
Note All types are represented by a icon.

4 To get a feel for what's particular to this behavior, click on the Subtype tab on the Property pane.

This displays those properties specific to the selected type. For MouseDriver, this is Controlling Window, Forward Axis, Speed Feedback, and Steering Feedback.

5 Select and drag the MouseDriver onto the Behavior Root task in the Task Scheduler.

The Behavior Input Definition dialog box appears.



Take a moment to look over this dialog. You use this dialog only to assign inputs to the selected Behavior. There is, however, additional information to help you with your decision. For more information on the Behavior Input Definition dialog, see Chapter 14, The Behavior System.

Our MouseDriver-1 Behavior needs to know exactly what geometry to control with the mouse. For our tutorial, we will be controlling the VehicleAvatar.

To add the VehicleAvatar to the MouseDriver's InputList 1

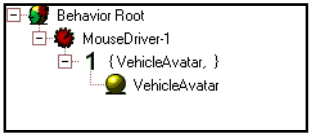
1 Select VehicleAvatar from the list of Movable objects and click the button.

2 Click OK.

In the Task Scheduler Pane, you should now see MouseDriver-1 as a child of the Behavior Root.

3 Expand the MouseDriver-1 behavior by clicking on the plus symbol to the left of it in the Task Scheduler pane.

This reveals a 1 which represents the MouseDriver's Input List. To the right of the 1 is {VehicleAvatar}, which is the actual contents of the input list, and contains a reference to VehicleAvatar, which we just added. Your Behavior Graph should now look similar to the following figure.



You have now successfully created a new behavior which says "Every Frame, execute the MouseDriver Behavior on the VehicleAvatar". Good job.

But wait! Before we can test it, we must make sure the MouseDriver needs no additional crucial information to operate. To know this, we must revisit the properties we examined in Step 2.

4 Making sure the MouseDriver-1 object (not the type, and not the input list) is selected, click on the Important tab on the Property pane.

The Important tab shows you just which properties the behavior author felt were crucial to getting the behavior up and running. For MouseDriver, the only crucial property is the Window object in which our behavior will be active (Controlling Window).

5 Since we want it to run in the Development window, double click on the Controlling Window Property and select DevWindow-1 from the list of Window objects.

We are now ready!

To test your MouseDriver behavior

1 Click the Run in DevWindow button to run your simulation in the Development Window.

2 Put your mouse in the Development window and observe what happens.

Wow, bet you lost the VehicleAvatar, right? Well, don't worry, in Lesson 3 we'll show you a quick way to fix that. For now, stop your simulation and find your van using the navigation and placement techniques you learned in the previous tutorials. (What? You didn't do the previous tutorials?) Once you've found it, you might want to put it back on the terrain somewhere. An easy way to do this is to select the VehicleAvatar node and set it's translation property to (0,0,0).

Behaviors can be dangerous to your scene arrangement. Now is a good opportunity to save your project.

Note If you loaded the template tutorial1-3.up at the beginning of this tutorial, save your current project with a different name. We suggest saving frequently so you can always come back to your simulation when it was in a "nice" state.

Lesson 2 Summary

The MouseDriver Lesson gave you a feel for how to create and schedule a Behavior, as well as how to identify crucial properties and how to edit them. Most Behaviors you will encounter will have 1 or 2 crucial properties and a few additional properties. Together these properties allow you to customize the way a Behavior works.

Lesson 3: Attaching the Viewpoint

Now that you know what you are doing, we're going to add few more behaviors to make your van move more realistically, as well as prevent you from losing it.

To prevent the van from just speeding off, we're going to attach our viewpoint to it. This will give us the feeling that we're actually in the van, instead of getting left in the dust. To do this, we'll add a Tether Viewpoint behavior.

To add a Tether Viewpoint behavior

1 Begin by starting where we left off in Lesson 2.

2 With the Behavior Workview Active, locate the TetherViewpoint type and select it.

3 Schedule it by dragging it onto the Behavior Root just like you did with MouseDriver-1.

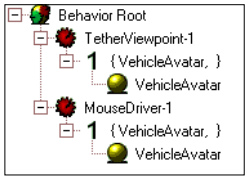
Again the Behavior Input Definition dialog pops up.

4 Add the VehicleAvatar to the InputList 1 of the Tether Viewpoint behavior, just like we did with the MouseDriver Behavior.

Note the description for TetherViewpoint tells us this will "tether" the viewpoint to the VehicleAvatar (input1).

5 Click OK to accept.

In the Task Scheduler Pane, you should now see TetherViewpoint-1 as a child of the Behavior Root and below MouseDriver-1.



Again, you can confirm by expanding the TetherViewpoint icon that the VehicleAvatar is indeed on TetherViewpoint's InputList 1.

Good Job! You have now successfully created a TetherViewpoint behavior. Our simulation now says, "Every frame, tether the viewpoint to the VehicleAvatar and control the VehicleAvatar with the mouse." But wait! Which viewpoint? As mentioned in the introduction to this lesson, we should check out this Behavior's Important properties. Of course, we know how to do this! We did it with the MouseDriver.

6 As a refresher, make sure the TetherViewpoint object is selected and then click on the Important tab in the Property pane.

Aha! We see our question, "Which Viewpoint?" is answered in the Property Pane. Filling in TetherViewpoint's "Viewpoint" property answers it. That's easy!

7 Double-click on the Viewpoint property and select Viewpoint-1 from the pop-up list.

We're ready to go!

8 Click the Run in DevWindow button to run your simulation in the Development Window.

That's much better than getting left behind! There is one small complaint, however. We're right in the middle of the van (we call this the "view from the transaxel"). If this view is less than pleasant for you, don't sweat it. The TetherViewpoint author has kindly provided an Offset Property.

9 With your simulation running, try adjusting this offset vector by double-clicking on it in the Property pane and modifying the Y and Z values

Adjust the sensitivity slider to Slow on the vector editor for this maneuver.

Finally, note that the viewpoint seems to move a little unevenly. This is because we are tethering the viewpoint before updating the van with the mouse, so we are receiving a one frame delay.

10 To fix this, move the MouseDriver-1 behavior so that it gets executed before TetherViewpoint-1 by dragging it onto the Root Behavior.

The rules of drag and drop are the same here as in the Scene Graph as learned in the earlier lessons.

11 Save your Project file if you haven't done so recently.

Lesson 3 Summary

Tethering the viewpoint is a classic behavior and a requirement for many simulations. Chances are you'll be using this behavior over and over again in different situations from Walkthroughs to Flyovers. By attaching the viewpoint to the VehicleAvatar, we made a sort of simple Avatar. For more information on viewpoint control and Avatars in general, refer to the Avatar User Manual.

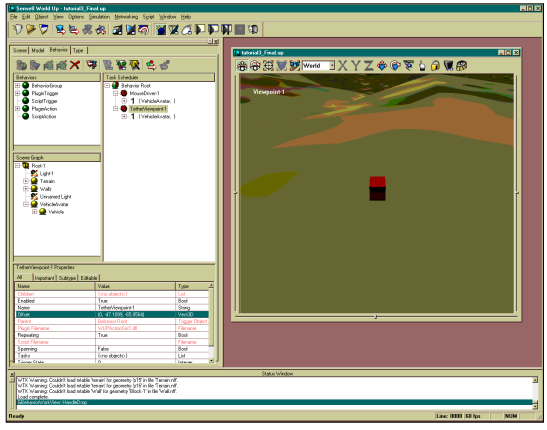
Lesson 4: Adding Terrain Following and Collision Detection

Our van is moving nicely now. There are, however some key laws of physics we seem to be ignoring, including gravity and rigid object permeability. In this lesson, we're going to get realistic about our simulation and address these problems by implementing terrain following and collision detection with our handy Behavior library.

To begin

1 Begin by starting where we left off in Lesson 3.

Your scene should look similar to the following figure.



2 Schedule a TerrainFollowLand behavior by dragging it onto the Behavior Root as we did with MouseDriver-1 and TetherViewpoint-1.

This brings up the Behavior Input Definition Dialog.

Note the TerrainFollowLand description. These descriptions are provided by the Behavior author, and seek to help you make a decision regarding what input we should use. In this case, as before, the "input object" is our van, so add the VehicleAvatar to the TerrainFollowLand's input 1 list.

3 4. Click OK and confirm the VehicleAvatar is in the input list as before.

4 Review TerrrainFollowLand's Important properties.

GroundObjectRoot specifies the ground. In our case, it is the group node that contains all of our terrain segments. (If you haven't grouped all of your terrain objects under a single group node, now would be a good time.)

5 Specify this group node as the Ground by double clicking the GroundRootObject property and selecting your terrain (or whatever you named it) group node from the list.

Great work.

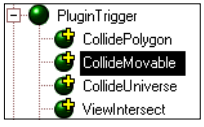
6 Click the Run in DevWindow button to test your new terrain in the Development Window.

The TerrainFollowLand author saw fit to give you some flexibility in deciding just how far above the terrain you wish to follow.

7 Try increasing TerrainFollowLand's DistanceOffGround property (while the simulation is still running).

Now that our van is happily obeying the laws of physics, let's handle the law of brick walls: collision detection.

8 In the Behavior Workview, take a moment to look at the CollideMovable Trigger, which is located under the "Plugin Triggers" section.



A Trigger is just like an action, except that it only fires (that is to say, executes it's children) if a condition is satisfied. In this case, CollideMovable fires if our van collides with a wall. For this to happen, we can guess our CollideMovable will need two inputs – One for our van and one for the list walls. Let's see how this works.

To create a CollideMovable Trigger

1 Drag a CollideMovable trigger onto the Behavior Root, just as we did with the other Behaviors.

2 In the Behavior Input Definition dialog, notice that this Trigger takes two inputs, unlike our previous actions (just store that fact in the back of your mind). This fact, combined with the description ("Collide movable checks for intersection between two user specified geometries") confirms our suspicions that we'll need two inputs for this Trigger to work.

3 For CollideMovable's Input 1, select the VehicleAvatar.

4 For CollideMovable's Input 2, select all four wall objects.

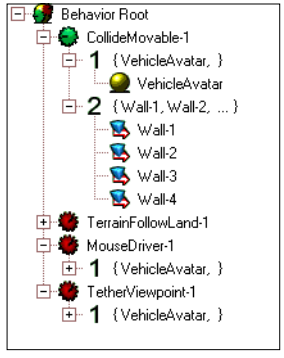
Do not select the group node (if any) that they may be under.

5 Click OK.

6 Look at CollideMovable's inputs in the Task Scheduler by expanding Collide Movable.

Note there are two inputs here, not just one.

7 Visually verify these are correct before proceeding.



8 Now, before testing our new collision detection, check to see if it had any important properties.

9 Click the Run in DevWindow button to test your new collision in the Development Window.

What happened!? Don't worry. The CollideMovable trigger isn't broken. In fact, it's working perfectly fine. We just don't know it because we didn't tell it what to do when the van ran into the wall. To remedy this, let's tell CollideMovable-1 what to do when the van does run into the wall.

10 Locate the BounceBack action from your list of Behaviors.

Now, before creating this action, think! We want our van to bounce backward only when we run into the wall, not every frame. So, instead of dragging this action onto the BehaviorRoot, we're going to drag it onto the CollideMovable-1 Trigger. Got it? (If not, there's more information to be had in the Behaviors section of this manual). Okay, go ahead and do it.

11 Drag the BounceBack action onto the CollideMovable-1 Trigger.

In the Behavior Inputs, we note by the description that BounceBack is going to bounce input 1 backward. For BounceBack's input, take a leap of faith with me here and check the radio box that says "Parent Out 1." Why not just add the van like we always do? Well, we could have, but that would have been less daring. Instead, what we said by checking "Parent Out 1" is, in essence, "Mr. BounceBack-1, please use the output coming from Mr CollideMovable-1."

For more information on this, see Chapter 14, The Behavior System. A quick check on BounceBack's Important properties reveals there are none, so let's roll. If all systems are go, you should be happily following the terrain and running into walls like any good physical object. Wait a minute, where's the fun in that?

Lesson 4 Summary

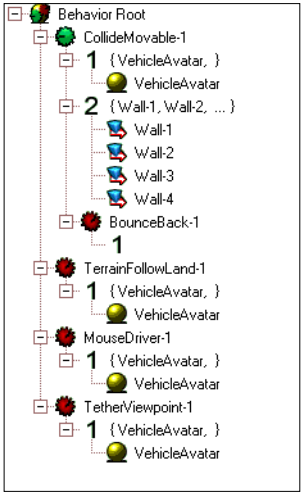
Like Viewpoint Tethering, Terrain Following and Collision Detection are classic simulation building blocks that you can use in a variety of settings. The crucial difference to understand in this Lesson is that Terrain Following is simply an action – that is to say, the van follows the ground every frame. Collision Detection also happens every frame – which is to say the system checks for a collision between the van and the walls every frame. However, not every frame does a collision occur.

A trigger can thus be though of as an event detection mechanism that, when detected, notifies others. In this case, BounceBack gets notified, which is what we want, since we only want to bounce back when we collide. This "If this happens Then do this…" logical structure is a cornerstone in any programming system, and now that you see how it works, you can quickly begin incorporating other such "high level" interactions.

Behavior Tutorial Review

As we have seen, the idea of a behavior covers a wide variety of activities. It can be useful to think of behaviors in terms of Triggers (or events) and Actions (or responses). Wiring these together forms a behavior. Consider as an analogy the human reflex. The Trigger is the event of the hammer on the knee. The response is the kicking of the leg. Together this event/response forms the reflex behavior.

The key to understanding Behaviors in WorldUp lies in the Behavior Scheduler, which is the network that tells you what happens when and to whom.



In our simple simulation, we scheduled behaviors off of the "root" of the graph, which means they were executed every frame. We also scheduled a behavior that was executed only when an event occurred (Bounce-Back-1). In addition to these combinations, there are several other basic combinations that allow you to achieve a wide variety of logical situations, including if – then, while –do, and begin - until. Chapter 14, The Behavior System will help you understand these concepts and more in greater detail, as well as show you how you can author your own behaviors!

For now, take a look at the remaining set of Triggers, and think about how these might be combined with the actions to form higher level behaviors. Start simply at first. For example, how about adding a PlaySound action to CollideMovable, so when the van hits the wall, it makes a crash?

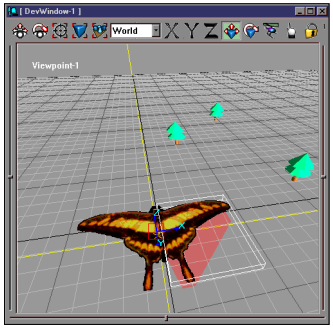
Tutorial 4: Paths – Your Doorway to Animations

In this tutorial, we're going to introduce another way to add interaction to your simulation: Paths. A *path* is a list of position and orientation values that can be applied to any WorldUp Movable or Viewpoint object. See Chapter 15, Paths for more information.

In this lesson,you will start with an imported model, adjust its center points, create simple and complex animations, and experiment with interpolation methods.

To begin

➤ Open the project file titled Tutorial4-1.up in the \Tutorials directory.



Adjusting the center point of the left wing of the Butterfly

Adjusting the Wing Center Points

Your butterfly is already assembled in a hierarchy, with the body as a parent of the left wing and the right wing. The wings are attached to the body, but their center points are still at the default position. We need the wings to rotate about the body, so the first thing we will do is to move their center points to be in the center of the body. Then, when we rotate the wings, they will rotate about the body, instead of around the center or the wing itself.

To adjust the wings

1 Select the body and zoom to it.

2 Select the Object Translate button at the top of the DevWindow.

You can only adjust the center points of objects in this mode.

3 Select the right wing by clicking on it.

4 Hold down the SHIFT key.

You will see a red square appear. This square represents the object's center point.

5 Drag this until it is over the center of the Butterfly's body.

6 Repeat for the left wing.

7 Save the Project.

Lesson 1: WingFlap Animation

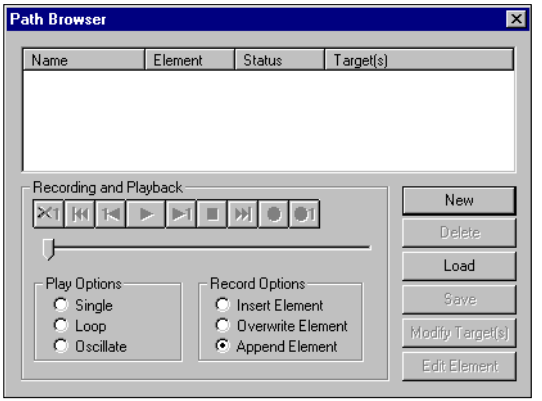
To animate the wing

1 Select the body and click the Zoom All button on the Development window toolbar.

2 Setup your viewports so that you can see what you are doing from all sides.

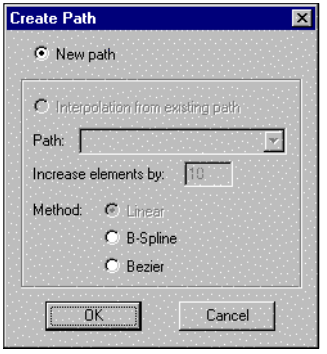
Select the last choice in the Display Options dialog box.

3 Bring up the Path Browser by clicking on the Browser – Toggle Path button on the toolbar, or by selecting Path Browser from the View Menu.



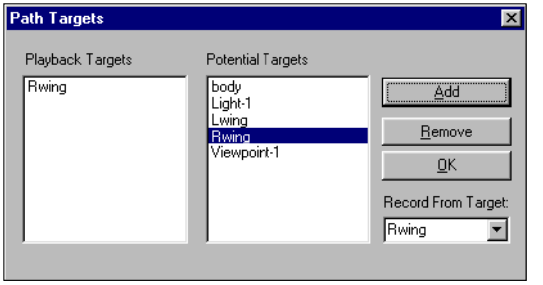
4 Create a new path by clicking the New Button.

You will see the following dialog, asking whether you want to create a new path, or an interpolated path.



5 Select New path, and click OK.

The system will automatically name it Path-1 for you, with a corresponding file name of Path-1.PTH. No actual file exists until you explicitly save your path. You should always rename it to something that makes sense to you before saving the path, otherwise you end up with multiple Path-1.PTH files, and may become confused as to which is which.



6 Double-click on the Target column of your new path to set a Record From Target.

Since this is an animation of the right wing, select that as your target, and Click the Add Button. You will see that the Rwing shows up as a Playback Target and as a Record From Target for this path.

7 Click OK.

You should now see the Target set to Rwing in the Path Browser. Your path has zero elements, because it is still empty.

We are now ready! Lets record the first frame of our animation right here, with the wings in the open position.

8 Click the Record 1 Frame button on the Path Browser to record the first frame.

Note that the number of elements in the path is now 1. Whee!

9 Right-click on the right wing in the Scene Graph pane, and select Position Object.

10 Pick the Orientation tab, and move the roll slider until the wing is in the fully-flapped position.

Try a Roll of -80.

11 Click OK.

12 Click the Record 1 Frame button on the Path Browser to record your second frame.

Note that the number of elements in the path is now 2. This is a very simple path, so this is the last frame. Our path is done.

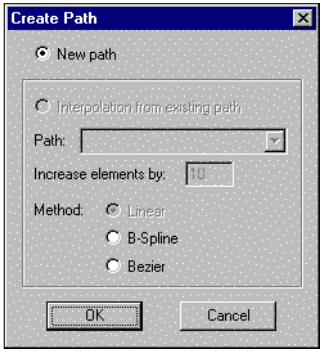
13 Try playing the path by clicking the Rewind button on the Path Browser, then clicking the Run button.

The path plays once and stops. To play it again you will have to rewind it. Set the path to loop and press play. Now the path will play indefinitely, and you can get a good idea of what your flapping animation will look like. It's a little too fast and jerky.

We can smooth it out and slow it down by interpolating the path.

14 Click New on the Path Browser.

15 Select Path-1 from the pull-down menu in the Create Path dialog box.



16 Type **5** into the Increase elements by box.

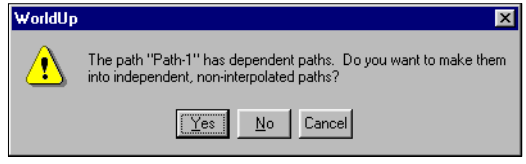
This inserts five elements between each of our old elements, which will make the path smoother and slower. Since one path element is played per frame, we have increased the duration of our path from two frames to six frames, so the butterfly flaps 1/3 as fast.

17 Click OK.

18 Set the playback Target of your new path to Rwing to see it play back, similar to Step 6.

19 Click Add and then OK.

Now you can delete Path-1 since we don't really need it. We just created it to interpolate from. We will be prompted to save Path-2 as an independent, non-interpolated path now.



20 Click Yes.

21 Rename your path to RwingFlap.

To do this, go to the Type Workview, find the Path Type and expand it, locate your path and edit its Name property.

22 Now repeat these steps to flap the left wing, except set the roll to +80 in the second element, instead of –80.

When you are done, you should have two paths, called RwingFlap and LwingFlap. You can play them together, by holding down the CTRL to select the names of both the paths, then clicking Play. Pretty cool.

23 Save the project.

This prompts you to save your paths as well. That's a good idea, lets do it.

You have a flapping butterfly, congratulations. Select the body and you can have fun making your flapping butterfly fly around by dragging it with the mouse.

Lesson 1 Summary

The wing flap path Lesson gave you a feel for how to create and playback simple and interpolated paths, and how to rename them.

Lesson 2 A Longer Animation

Now a butterfly flapping its wings is a wonderful thing to behold, but even cooler would be to see it fly around in that carefree way that butterflies have! We will record a longer animation of the butterfly flitting about the world. For this animation we rely on the fact that the wings are children of the body. We can animate the body and the wings will fly around attached to it. Since this animation is more complicated, I made a helper structure out of blocks first, to help me get the orientation of the butterfly right, so that it is always facing in the direction that it is flying.

Import the world tutorial4-3.up, and note the blocks that I have positioned for the path I am making. Each block represents the position of the Butterfly for one path element. The distance between the blocks determines the relative speed of the butterfly. If you wanted your butterfly to fly at a constant speed, you would need to place your blocks exactly the same distance apart. I used uneven spacing, so that at times the butterfly is speeding up, and at times it is flying slower. This makes the motion look more realistic.

1 Create a new path (non-interpolated) and set the Target to Body.

   In the scene graph, it is helpful to lock the body as the selection using the Lock Selected  button, so you don't deselect it accidently.

2 Move the Body so that it is over the first block in the Development window, Block-0.

3 Orient the Body to look towards Block-1.

4 Record one frame.

   In the Property pane with the Important tab selected, you can view the Translation and Rotation property values for the frame you just recorded.

   The values don’t have to be exactly the values shown in this Tutorial, but something close to them means you're on the right track in creating your animation.

   You should have a Translation of (0.000000 0.000000 0.000000) and a Rotation of (0.000000 0.000000 0.000000 1.000000).

5 Move the Body to Block-1 (using the Rotate Object tool or the positioning dialog) and orient it (using the Rotate Object tool or the positioning dialog) to look towards Block-2.

   Note Remember you can right-click and select Position Object with your cursor on Body.

6 Record one frame.

   You should have a Translation of (0.470277 0.000000 2.365093), and a Rotation of (-0.125140 -0.270752 0.053865 0.952960)

7 Move the Body to Block-2, orient it to look towards Block-3, and record one frame.

   You should have a Translation of (4.834613 -8.900558 8.266823), and a Rotation of (-0.262064 -0.734272 0.144990 0.609216).

8 Move the Body to Block-3, orient it to look towards Block-4, and record one frame.

You should have a Translation of (7.420220 -12.141972 5.765294), and a Rotation of (0.058921 -0.793465 -0.085788 0.599651).

9 Move the Body to Block-4, orient it to look towards Block-5, and record one frame.

You should have a Translation of (20.523592 -9.167504 3.858362), and a Rotation of (0.023860 -0.960125 0.107181 -0.257103).

10 Move the Body to Block-5, orient it to look towards Block-6, and record one frame.

You should have a Translation of (16.338177 -12.147344 -5.011420), and a Rotation of (0.038578 0.763583 -0.238163 0.598942).

11 Move the Body to Block-6, orient it to look towards Block-7, and record one frame.

You should have Translation of (4.556245 -16.004597 -9.549655), and a Rotation of (-0.086254 -0.888132 -0.025451 -0.450703).

12 Move the Body to Block-7, orient it to look towards Block-8, and record one frame.

You should have Translation of (2.516228 -13.548829 -12.062134), and a Rotation of (0.198164 0.361845 -0.143863 0.899501).

13 Move the Body to Block-8, orient it to look towards Block-9, and record one frame.

You should have Translation of (-9.066780 -8.501297 -2.413167), and a Rotation of (0.253034 -0.616815 -0.319437 0.673404).

14 Move the Body to Block-9, orient it to look towards Block-0, and record one frame.

You should have Translation of (-0.304579 -1.260750 0.896567), and a Rotation of(-0.082994 -0.631824 0.012978 0.770547).

15 Move the Body to Block-0, orient it to look towards Block-1, and record one frame.

You should have Translation of (0.000000 0.000000 0.000000), and a Rotation of (0.000000 0.000000 0.000000 1.000000). This is the last frame.

16 Play back the path using the play option of your choice and check how you like it.

If anything looks wrong, step through the elements one by one using the Play1 button. If you want to modify one of the elements, step to it and select the edit Element button. You can then reposition the body as you like and re-record the frame.

Our rough path is recorded, now we will interpolate it like we did with the flapping paths in Lesson One. This time lets try all three of the interpolation types to see how they are different.

17 Make a Linear interpolation, a Bezier interpolation, and a B-Spline interpolation of your path. Use the default value of 10 elements.

18 Rename your paths in the Type Workview, before you forget what they are.

Name them to FlyAround Bezier, FlyAround Linear, and FlyAround B-Spline, respectively.

Now, let's compare the paths.

19 Select the body node.

20 Right-click and select Duplicate Tree to clone the butterfly.

A Duplicate Attachments dialog box appears.

21 Click Create New Path in the dialog box and click OK.

22 Repeat to make a third butterfly.

23 Open the Path Browser.

24 Assign each FlyAround path to a different butterfly body, and start all of the butterflies flapping.

25 Now select all of the FlyAround paths, and select Play 1 to let you step through and see how they are different.

Or, just click Play. Pick the one you like the best to use in the Vehicle simulation.

26 Save the project and all of your paths.

Lesson 2 Summary

Making a complicated animation can be challenging, and this example shows you a few tricks to make it easier. We also learned how to find and edit elements as desired. We learned about the effects of different types of interpolation.

7

Using the Workviews

This chapter provides a description of the overall Project Workview and each individual Workview.

What are Workviews?

The new Workviews provide a more efficient work flow by simplifying the user interface, using tabs, to focus on the current task. The Project Workview refers to the overall tabbed interface that brings often-used features of WorldUp into your view.

The term *view* in this case should not be confused with a visual view of the simulation, but is rather an organizational/informational view of a certain aspect of the project required to perform a particular type of work in WorldUp.

The Project Workview

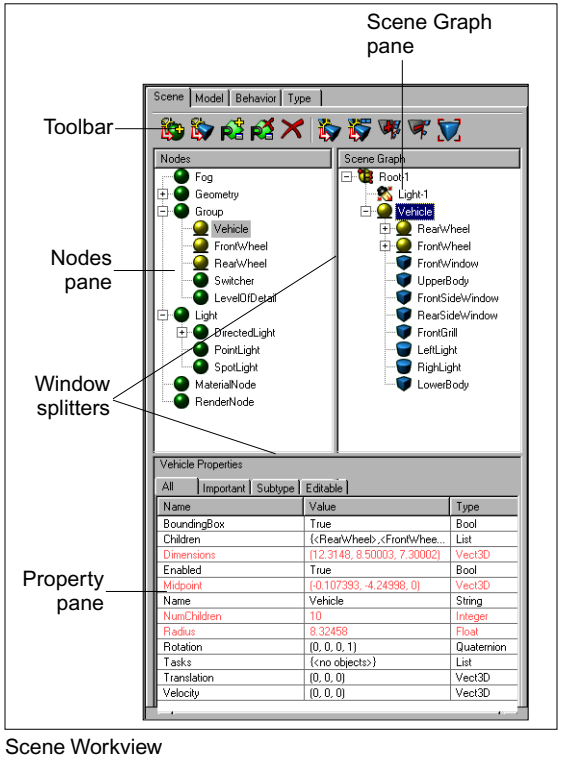
The Project Workview has a row of tabs along the top that allow you to select a certain *view* of the project. These tabbed views are the Scene, Model, Behavior and Type Workviews.

The Project Workview displays a single Workview at a time selected by the tab. Each Workview is a collection of sub-windows that display information necessary to accomplish a specific type of task in WorldUp. These nested subregions are called *panes* to differentiate them from windows. A window can

be moved independently, while a pane is fixed in a position within a window, possibly resizable using a splitter.

Scene Workview

The Scene Workview is the most commonly used workview for creating new objects, editing their properties, assembling the scene, and laying out the scene graph. The Scene Workview consists of the Nodes pane on the left, the Scene Graph pane on the right, and the Property pane at the bottom.



Scene Workview

The Nodes pane shows all the Node types, which are the types that can be added to the scene graph, and their instances. Here you can create new objects by clicking on the Instantiate Selected Type button or by dragging and dropping the type from the Nodes pane to the Scene Graph pane.

The supported Node types are geometries of type Block, Cone, Cylinder, Imported, Sphere and Text3D; Lights of type DirectedLight, SpotLight and PointLight; Groups of type Group, Switcher, and LevelOfDetail with other types such as Fog, MaterialNode and RenderNode.

The Scene Graph pane displays the scene graph, which shows how the objects are organized in your scene. You can rearrange the objects in a scene graph using the drag and drop method.

Note The following types are special in that they cannot have children: Fog, MaterialNode, and RenderNode. Also, the Geometry and Light types are abstract which means they cannot be instantiated. You use their derived types instead.

The Property pane is helpful for viewing and changing the properties of an object. The Property pane has four tabs:

All – Shows all the properties of the selected object. The properties shown in Red are protected or read-only which means they cannot be changed.

Important – Shows the most specific and basic properties of an object which are required to create the object.

Subtype – Shows all the properties that exist in the derived type that don't exist in its parent.

Editable – As the name says, lists all the properties which can be edited.

You can create custom subtypes of any of the Node types in this Workview, adding properties as necessary. You cannot add or delete properties for the built-in types. You can delete properties that you have added to your custom types.

The Toolbar contains icons that represent commonly used commands.

Create Subtype – Creates a subtype of the selected type in the Nodes pane. You can customize subtypes by adding new properties.

Instantiate Selected Type – Creates a new object based on selected object type in the Nodes pane. Adds the new object under the Type in the Nodes pane and to the end of the scene graph in the Scene Graph pane. You can also drag-and-drop a Type from the Nodes pane to the Scene Graph pane.

Add Property – Adds a new property to the selected subtype. The Add Type Property dialog appears to enter the name, type and initial value for the new property. New properties can be added only to subtypes, not to built-in types.

Remove Property – Deletes properties of selected subtype. Only properties added to a subtype can be removed.

Delete Selected – Deletes the selected object. You can delete multiple objects at once by selecting the objects while holding down the CTRL key. You can also use the DELETE key to delete objects from the scene graph.

Duplicate Selected – Copies the selected object.

Duplicate Selected with Children – Copies the selected object and all of its children.

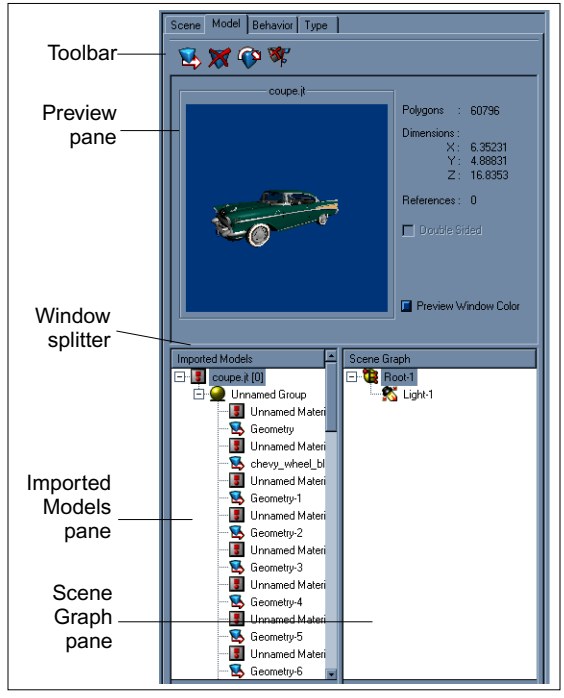
Expand Selection – Expands the scene graph tree to display selected object's children.

Collapse Selection – Collapses the selected object's tree.

Zoom to Selection – Zooms to the selected object in the Development window.

Model Workview

The Model Workview is the center for importing models into a WorldUp simulation.



Model Workview

The process of importing objects is first loading a 3D model in the Model Workview, previewing the model, and then adding the model to the scene graph. The loading process allows you to scale the geometry, combine all the geometries, or make the midpoint of an object its center.

The Preview pane displays the loaded object and its properties, such as number of polygons, dimension, etc. To add a loaded object to the scene, you drag and drop the whole object or parts of it to the desired location in the scene graph.

For more details on how to import models, see "Importing Models from Third Parties" on page 105.

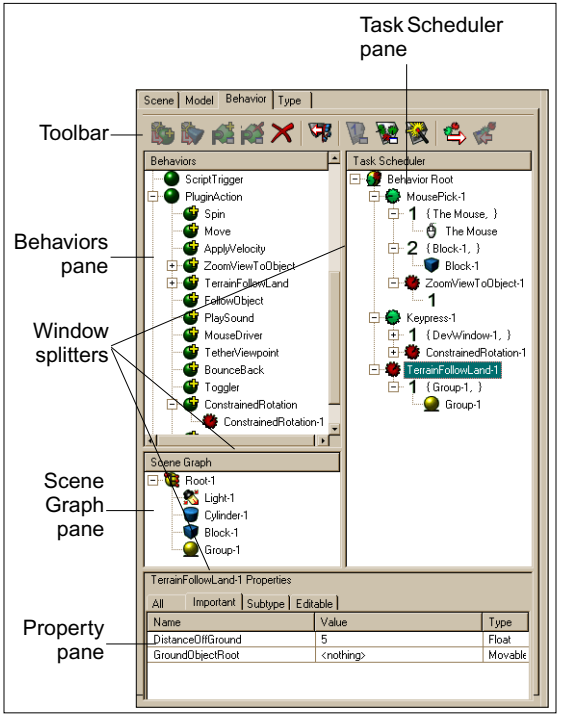
The Toolbar contains icons that represent commonly used commands.

- Import New Model – Imports a new model into the project.
- Remove Model from Project and Memory – Removes the selected model from the project.
- Reload Model – Reloads the selected model if it has been modified without having to reload the project.
- Unreference Model – Deletes all objects from the scene that refer to the selected model. The model remains in the Model Workview.

Behavior Workview

The Behavior Workview is where all behavior creation and scheduling is done. The Behavior Workview contains four panes which together allow you to create a new behavior, schedule the behavior, add inputs to the behavior, edit the behavior's properties, and finally export the behavior for re-use.

For more details on adding behaviors to your scene, see Chapter 14, *The Behavior System*.



Behavior Workview

The Behaviors pane lists all of the registered Behavior types that you can create. The three key types of behaviors are Group, Trigger, and Action.

The Task Scheduler pane shows the execution flow of all scheduled behaviors, as well as each behavior's inputs. This allows you to both see the execution order and to drag-and-drop objects onto specific inputs.

The Scene Graph pane provides you with a source of objects from which to drag onto a behavior's inputs.

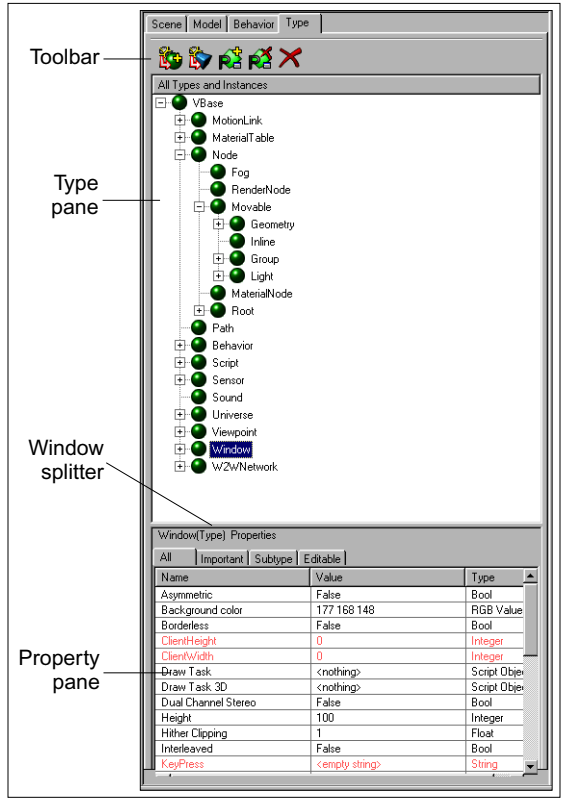
The Property pane allows you to quickly identify and modify a behavior's parameters. (Use the Subtype tab for a quick view of properties specific to a certain behavior.)

The Toolbar contains icons that represent commonly used commands. In addition to the commands available in the Scene Workview, the Behavior Workview contains the following icons:

- Unschedule Behavior – Removes a behavior from the Task Scheduler.
- Edit Behavior Scripts – Opens the Script Editor and displays the contents of the selected script.
- Edit Inputs – Allows you to edit the behaviors inputs and outputs.
- Behavior Wizard – Opens the Behavior Wizard dialog box.
- Import Behavior – Imports a behavior script.
- Export Behavior – Exports the current behavior for use later.

Type Workview

The Type Workview allows you to create custom subtypes, viewing and changing type and object properties. The Type Workview consists of a Type pane and a Property pane.



Type Workview

The Type pane displays the complete set of WorldUp types, including non-Node types such as Windows, the Universe, Sensors, and MotionLinks.

The Property pane shows the properties of the selected type or object in the Type pane.

The Toolbar contains icons that represent commonly used commands. These commands are explained in the Scene Workview as their functions are exactly same in Type Workview.

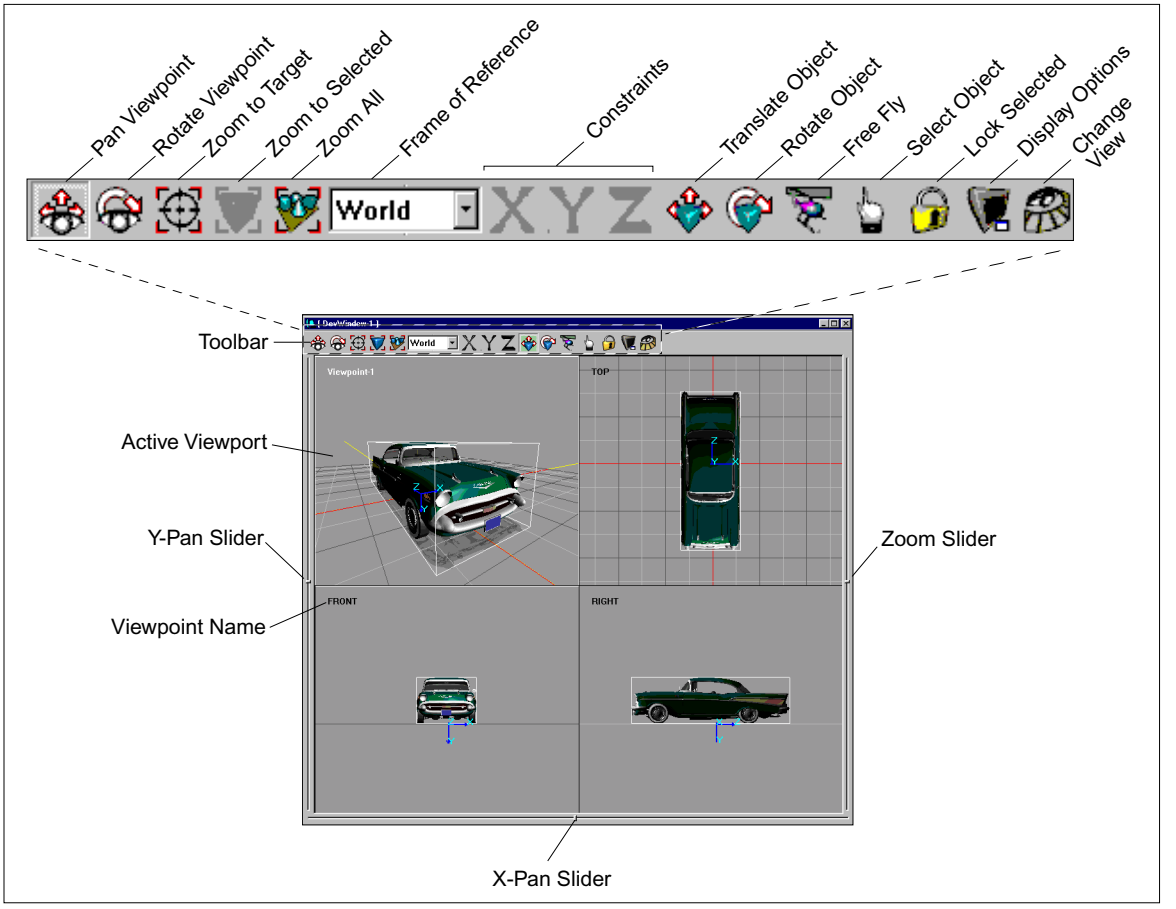
8

Development Window – Navigation and Manipulation

This chapter describes the various navigation and manipulation methods available for viewing your graphical objects and moving around your scene in the Development Window. This chapter also explains how to set up multiple viewports, change views, and customize display options, such as shadows and grids.

The Development Window

The Development window is where your scene is rendered. This is the window you use to interact with the 3D scene while authoring your simulation. The Development window and its various parts are shown in the following diagram.



The Development Window and Toolbar

The navigation and manipulation tools located on the Development window toolbar are described in detail in this chapter.

Moving Around the Simulation

This section discusses panning, rotating, and zooming in the Development window.

Pan Viewpoint

Clicking and dragging in the Development window with this tool selected translates (pans) the viewpoint in a horizontal and vertical direction. The amount of translation is in proportion to the proximity to the viewpoint target.

The viewpoint target is a virtual point projected along the z-axis of the viewpoint. When WorldUp is first started, this viewpoint is centered at the

universe origin. Panning, rotating, and zooming the viewpoint changes the position of the viewpoint target.

Rotate Viewpoint

Clicking and dragging in the Development window with this tool selected rotates the viewpoint about an axis formed in the direction of the world y-axis through the viewpoint target. Up and down dragging results in a rotation about an axis formed in the direction of the local x-axis of the viewpoint through the viewpoint target. Think of this in terms of a pole which has the viewpoint attached to one end and the other end is fixed to a pivot.

If an object is selected, the viewpoint target is the midpoint of that object. If no object is selected, the viewpoint target is the midpoint of the Root node, which is the bounding box center of all objects combined in the scene graph. Vertical rotation is limited to a 180 degree arc between looking straight down and directly up along the world y-axis.

This selection is disabled for orthographic views in order to keep the orientation of the viewpoint intact.

Zoom

In addition to the methods already described in this section, several controls allow you to zoom in on your viewpoint.

Zoom to Target

Clicking on an object when using this tool results in a smooth transition of the viewpoint to a new position that is closer to the point of intersection of the pointer and the target object. The amount of zoom is about 70 percent of the original distance.

This tool can be used to quickly navigate among readily visible elements in the universe. This tool is disabled for orthographic views in order to keep the orientation of the viewpoint intact.

Zoom to Selected

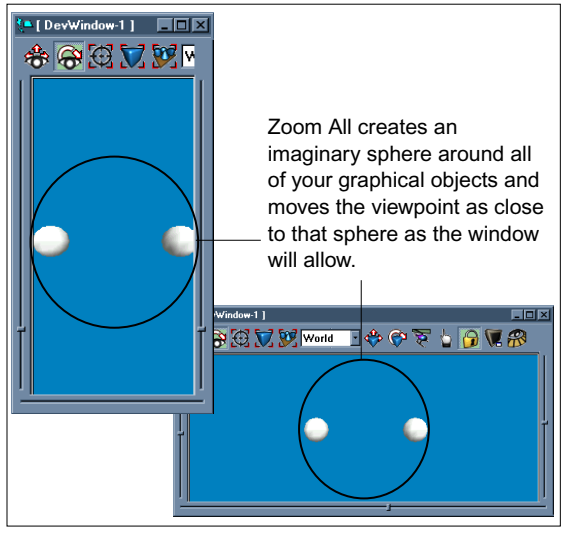
Zoom to Selected adjusts the viewpoint by fitting the window as close as possible to the bounding box around the selected node.

This option is disabled when nothing is selected or the object is a non-Movable. See Chapter 12, Editing 3D Objects for more details.

Zoom All

Zoom All works exactly as Zoom to Selected except the selection set is always the entire scene graph.

WorldUp adjusts the viewpoint by fitting the window to the smallest sphere that could be drawn around all the graphical objects in the simulation. Thus, the distance from the viewpoint to your objects depends on the height and width ratio of the window.



Free Fly

Free Fly allows you to navigate around the scene as if you are in a helicopter. Free Fly is a motion link that is connected only to the default perspective viewpoint.

This feature is disabled for orthographic views.

Manipulating Objects

Several tools are available for selecting, translating, and rotating objects in your scene.

Translate Object

Translate Object translates the selected object in the current reference frame. Holding down the left mouse button and dragging translates in the x,z plane passing through the center point of the selection. If multiple objects are selected, the translation plane is that of the first object traversed in a normal scene graph traversal.

Dragging the mouse with the right button held down results in a translation along the Y-axis in the selected reference frame.

You can select a new object with this tool by clicking on the object. Clicking while holding down the CTRL key allows multiple selections.

The *origin offset* of an object can also be moved using this tool. The origin offset defines the origin of the coordinate axes about which an object rotates. To move the origin offset of a selected object, hold down the SHIFT key while dragging. A red rectangle appears around the midpoint of the object and further dragging applies to the translation of the origin offset of the object.

Rotate Object

Rotate Object rotates the selected object in the current reference frame. Holding down the left mouse button while dragging up and down the node rotates the object clockwise and counter-clockwise about the selected rotation axis. This axis of rotation is displayed in red.

Holding down the right mouse button in this mode quickly selects another rotation axis in this order: x, y, z, repeat. Selecting the corresponding Constraint button produces the same results.

In this mode, new objects can be selected by clicking on them. Clicking while holding down the CTRL key allows multiple selections.

Constraints

Constraints allow you to define the axes in which to translate and rotate objects. By clicking on the x, y, or z button in translation mode, you lock translation to that particular axis. When rotating objects, this selects the axis to be rotated about. In rotate object mode, right clicking changes the axis to rotate about.

An axis constraint button is disabled whenever it is not possible to translate or rotate on that axis.

Frame of Reference

The Frame of Reference pull-down menu allows you to select a reference frame (coordinate system) about which objects are translated and rotated. Four reference frames are available:

- *World* – Object manipulation is performed in the universe's reference frame.
- *Parent* – Object manipulation is performed relative to the frame of the node's parent. If the parent is the Root node, this selection works exactly the same as *World*.

- *Local* – All manipulations are relative to the selected node's local frame.
- *View* – Object manipulation is performed in relation to the viewpoint's frame.

See "Coordinate Systems" on page 13 for more details on reference frames.

Select Object

Select Object allows you to select different objects in the scene, if Lock Selected is not enabled. To select multiple objects, hold down the CTRL key.

Lock Selected

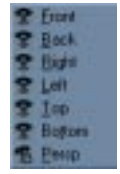
Lock Selected locks the current set of selected objects. The selection set can include graphical and non-graphical objects.

Using the Window Sliders

The three slider controls on the left, right, and bottom edges of the Development window allow you to control the viewpoint's current position along each axis.

- *Left slider (Y axis)* – Drag to move your viewpoint up or down.
- *Right slider (Z axis)* – Drag to move your viewpoint forward or backward (zoom in or out on the scene).
- *Bottom slider (X axis)* – Drag to move your viewpoint left or right.

Changing Views



The Change View button on the toolbar displays a pull-down menu allowing you to select a viewport for the active viewport.

The icon on the pull-down menu indicates *orthographic* view and the icon indicates *perspective* view.

To change your view

1 Click the Change View button.

2 Select one of the views on the pull-down menu for the active viewport.

Perspective mode is a three-dimensional view of the universe. Orthographic mode is a two-dimensional view of the universe. You may find orthographic mode helpful for positioning objects.

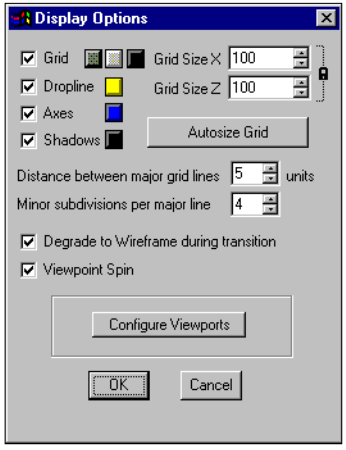
Setting Display Options

You can change the appearance of the panes in the Development window using the display options.

To set the display options

1 Click the Display Options button.

The Display Options window appears.



The *Grid*, *Dropline*, *Axes* and *Shadows* check-boxes allow these features to be toggled on and off. The adjacent colored buttons show the currently selected colors for each feature.

The Grid is composed of major and minor lines. The buttons next to Grid are for assigning the color for major grid lines, minor grid line, and the world axis line respectively. Clicking on any of these buttons brings up a standard color selection dialog box allowing you to select new colors.

Grid Size alters the dimensions of the grid in each direction. Clicking on the padlock icon to the right of the Grid Size text boxes allows you to lock one value to the other so that the grid is always square.

Autosize Grid evaluates the dimensions of the universe and creates a reasonable grid. The divisions and units do not change.

Distance between major grid lines specifies the distance in units between major grid lines.

Minor subdivisions per major line specifies the number of minor subdividing lines between each major grid line.

Degrade to Wireframe during transition, when checked, causes rendering to be performed in wireframe mode. Transitions and object manipulations, by default, degrade rendering to wireframe mode for better visibility while in use. To turn this off, uncheck the check-box.

Viewpoint Spin, when checked, keeps rotating the viewpoint when dragged and left in viewpoint rotate mode.

Configure Viewports allows you to select the number of viewports to display in the Development window.

Setting Multiple Viewports

Viewports allow you to set up multiple panes in a single Development window in which you render different views of the scene. This feature helps you to position and align objects in the scene. Each viewport can be set up to have a custom perspective view or built-in orthographic views of front, back, right, left, top, and bottom.

When using multiple viewports, you select one viewport at any given time on which all the navigation happens. This viewport is called the *active* viewport and is highlighted with a white border.

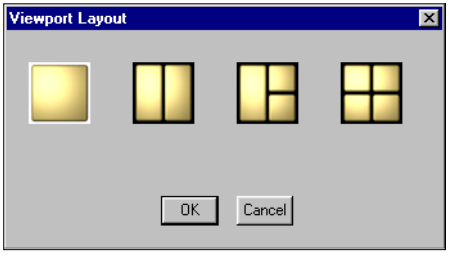
Manipulation of an object affects that object in all the viewports since rendering happens in all the viewports. Navigating with a viewpoint only affects the viewport to which it is attached. If the same viewpoint is attached to more than one viewport, the navigation changes the views of all those viewports. A viewport can be set as the active viewport by clicking anywhere in the pane.

To configure multiple viewports

1 Click the Display Options button.

2 Select Configure Viewports on the Display Options window.

The Viewport Layout dialog box appears.



3 Select the viewport layout you wish to use.

4 Click OK.

The selected viewport arrangement is created in the Development window using standard viewpoints. You can change the viewpoint in each viewport using the Change View button as described in "Changing Views" on page 83.

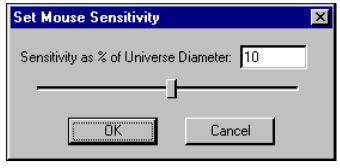
Setting the Mouse Sensitivity

You can control the speed at which you navigate through a simulation by specifying the percentage of the universe's total size that you want to cross with each mouse click.

To set the mouse sensitivity

1 Click the Mouse Sensitivity button.

The Set Mouse Sensitivity dialog box displays.



2 Drag the slider or enter a specific number in the text box to set the desired percentage.

For example, if your current setting is 1, entering 10 enables you to navigate ten times faster than your current speed. Each mouse click moves you a distance equivalent to 10% of the size of the universe.

3 Click OK.

9

Objects and Properties

This chapter defines objects and properties, and describes how to create, delete, duplicate, and find objects

Objects

Objects are one of the core building blocks upon which your simulation is built. Graphical objects are the objects that you can see in the Development window. The objects in a scene/simulation are what ultimately get drawn. Objects may be created or imported.

Creating an Object

You create objects from types in the Type Workview or the Nodes pane of the Scene Workview. Types serve as templates that contain the properties required to create a particular object. Thus, when you create an object, it contains all of the properties of the type from which it was created.

Note For some object types, tadditional methods are available for creating an object. These are discussed in the object type's related Workview chapter.

To create an object

1 In the Type Workview or the Nodes pane of Scene Workview, select the object type that contains the properties you want your new object to have.

Note If a pre-defined object type includes pre-defined subtypes, in some cases you cannot create objects from the parent type. For example, you cannot create objects from Geometry, but you can create objects from Block, Cylinder, Cone, Sphere, and Text3d.

2 Click the Instantiate Selected Type button.

This creates a new object of the selected type with a default name. You can modify the object's name later using the Name property in the Property pane.

For certain object types, WorldUp prompts you for specific information at this point. For example, creating a Sound object requires you to select the filename for the .WAV file you want the object to reference.

You can see your new objects in the Type Workview, indented under the type from which they were derived. If the objects that you created were descendants of the Node object type, the new objects also appear in the Nodes pane of the Scene Workview.

Deleting an Object

To delete an object

1 In the Type Workview or Scene Workview, select the object that you want to delete.

2 Press the Delete Selected button.

Duplicating an Object

You can copy an object by itself or with all of its children attached.

To duplicate objects

1 In the Type Workview or Scene Workview, select the object that you want to duplicate.

2 Click one of the following buttons:

- *Duplicate Selected* – copies the selected object only.
- *Duplicate Selected with Children* – copies the selected object, plus any child objects it may have. For more information on the scene graph and child objects, see page 23.

The duplicated object contains the same properties and property values as the object from which it was duplicated.

For information on modifying property values, see page 91.

You can also duplicate objects from the Scene Graph pane by holding down the CTRL key as you drag nodes within the scene graph. The new objects are automatically named based upon the names of the objects from which they are duplicated. You can later rename the duplicated objects.

Creating and Deleting Subtypes

You create subtypes from existing types. When you create a subtype, it inherits all of the properties of the type from which it was derived and becomes subordinate to that type.

The two cases in which you create a subtype are as follows:

- You have or are creating an object that will requires a user-defined property.

You cannot add properties to pre-defined object types, so you must create your own type to which you can add the property. Once you have created the type and added the necessary properties, you can create new objects from that type and they inherit those new properties, or you can drag existing objects of the same type onto the user-defined type, and those objects now inherit the new properties.

- You want to create multiple objects that require the same value for certain pre-defined properties.

  For instance, in your simulation you plan to import several different models of cars, and you will be attaching a script called RACE.EBS to each car. You could create a subtype of Imported called Race and set its Tasks property to RaceScript. Any models that you drag onto this type from the Resource Browser will inherit the RaceScript value for the Tasks property.

To create an object type

1 In the Type Workview or the Nodes pane of the Scene Workview, click the type that contains the same basic properties you want your new subtype to have.

  Note If a pre-defined object type includes pre-defined subtypes, in some cases you cannot create subtypes from the parent type. For example, you cannot create subtypes from Geometry, but you can create subtypes from Block, Cylinder, Cone, Sphere, and Text3d.

2 Click the Create Subtype  button.

  This creates a new subtype of the selected type with a default name. You can modify the subtype's name later using the Name property in the Property pane.

  You can see your new subtype in the Type Workview or the Nodes pane of the Scene Workview, indented under the type from which it was derived.

To delete an object type

1 In the Type Workview or the Nodes pane of the Scene Workview, select the user-defined type that you want to delete.

2 Click the Delete Selected  button.

Finding an Object

You can locate an object in your project using the Find Object command.

To find an object

1 Select Find Object from the Edit menu.

  The Find Object dialog box displays.

2 In the Name box, type the exact name of the object you are looking for and click OK.

  WorldUp selects the specified object if it exists.

Properties

You can add properties to user-defined object types only. WorldUp provides several pre-defined properties with each pre-defined object type, such as Name for all object types and Background Color for the Window object type.

For a description of all WorldUp properties, click WorldUp Contents from the Help menu and open the Properties topic.

User-defined properties are not used by WorldUp until they are referenced from scripts or used to trigger reactions to events. Scripts can reference any properties, pre-defined or user-defined. Likewise,

you can use both pre-defined and user-defined properties to trigger event reactions (though not all pre-defined properties generate events).

For more information on scripts and events, see the *WorldUp Programmer's Manual*.

When you select an object or object type, its properties are displayed in the Property pane. If you have multiple objects/types selected at once, the Property pane displays a collective list of properties for all the selected objects/types, but only displays a value for a property if it is a property and value that all the selected objects/types have in common.

An object's properties can be created, changed, or removed during run time. They can also be inherited from their parent type during run time, hence WorldUp supports *dynamic inheritance*. When the simulation is running, the changes you make to an object's properties take effect immediately. Likewise, any property changes that are made by scripts and behaviors are reflected in the Property pane, if you have enabled the Update Properties From Script option (see "Automatically Updating Properties" on page 92).

Adding a Property

You can add properties to user-defined object types only.

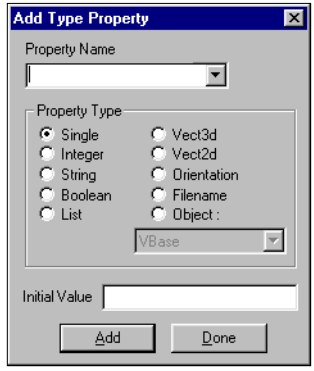
To add a property

1 In the Nodes pane of the Scene Workview, click the user-defined type to which you want to add the property.

Note You cannot add properties to pre-defined object types (that is, the WorldUp default object types). You can only add properties to user-defined object types. See page 88 for instructions on how to create an object type.

2 In the Scene Workview toolbar, click the Add Type Property button.

The Add Type Property dialog box displays.



3 Enter a unique name for the new property in the Property Name field.

You can see a list of all of the existing property names for the selected type by clicking the drop-down list.

4 In the Property Type section, click a data type for the property.

To choose an object type as the data type, click Object, then click on an existing object type from the drop-down list.

For a description of each data type, search for Data Types in the online Help.

5 Type a value in the Initial Value box.

6 This is the value that will initially be assigned to the property. You can later modify this value (see "Modifying a Property Value" on page 91).

Note You can leave the Initial Value box empty and the default value will be used. This value is dependent on the data type. For example, the default value for Boolean is False and the default value for an Orientation is 0, 0, 0, 1.

7 Click Add.

8 If desired, continue adding properties to the selected type in this manner.

9 When you are finished, click Done to close the Add Type Property dialog box.

The properties you created now appear in the Property pane for the selected object type, and for any subtypes or objects nested within that type. You can see a list of the user-defined properties for the selected type or object by clicking the Subtype tab of the Property pane.

Removing a Property

To remove a property

1 In the Type pane, click the object type whose property you want to remove.

2 In the Property pane, select the property you want to remove by clicking somewhere within the property row.

3 Click the Delete Type Property  button.

Modifying a Property Value

You can modify the value of any property that appears in black text in the Property pane. Properties that appear in red text are read-only.

To modify a property value

1 In the Type pane, click the object or object type whose properties you want to modify.

2 In the Property pane, click the tab that contains the properties you want to change.

- *All* – Lists all properties of the selected object or object type.
- *Important* – Lists the most specific and basic properties of an object which are used to create the object, such as Filename for Sound objects or Height for Block objects.
- *Subtype* – Lists only the properties of the subtype which are not present in the parent type.
- *Editable* – Lists only properties whose values can be edited at any time.

3 Select the property whose value you want to modify by clicking somewhere within the property row.

4 To modify the value of any editable property, click on the selected property in the Value column.

In most cases, this brings up an edit box in which you can type in the value.

5 When finished typing, press ENTER.

The new value replaces the old one in the Property Value column of the selected row.

Many property data types allow for alternative methods of modifying property values that you may find more convenient. The following table describes these methods for each data type.

Data Type	To modify the property value...
Boolean	Double click on the Name column of the property to toggle between True and False.
List	Double-click on the Name column of the property to display the Edit List dialog box. In the Objects box, click an object, then click Add to add it to the property's list. To remove an object from the list, click the object in the List box, then click Remove. To change the order of the objects in the list, use the Up and Down buttons. Click Done when you are finished.
Vect3D or Orientation	Double-click on the Name column of the property to display a dialog with controls to help you select new values.
Filename	Double-click on the Name column of the property to display the Open dialog box. Navigate to select the name of the file that contains the information you need and click Open.
Object	Double-click on the Name column of the property to display a list of all objects that are descendants of the type specified as the data type. Click the desired object, then click OK.
Material	Double-click on the Name column of the property to display the Choose Material dialog box. Click the desired color, then click OK.
LOD Ranges	Double-click on the Name column of the property to display the Level Of Detail Ranges dialog box. See "LevelOfDetail Nodes" on page 26 for instructions on using this dialog box.
RGB	Double-click on the Name column of the property to display the Color dialog box where you can choose a new color.
Integer, Single, String, or Vect2d	Double-click on the Name column of the property to open an edit box in which you can type in a new value and then press ENTER.

Automatically Updating Properties

Property changes made by scripts and behaviors can be updated in the Property pane automatically.

To automatically update changes made from scripts

1 On the Options menu, select File Access Settings.

The Settings dialog box appears.

2 Select the option called Update Property Changes From Script and click OK.

Editing Properties In-Place

Many properties can be edited directly in the Property pane. Read-only properties are displayed in Red and cannot be edited.

To edit properties in-place

1 Double-click on the Name of a property.

2 To edit the text value in-place, do a *slow* double-click in the Value column.

This is the same type of click you use to rename a file on your desktop or in an Explorer window. If the property cannot be edited in-place, its edit dialog appears.

3 Type in the new value and press ENTER.

You can also copy the value by pressing CTRL-C. Then you can paste the value by pressing CTRL-V into any other property that is of the same type (provided that it is not read-only).

Property Types and How They Can Be Edited

Property Type	Edit In-place	Edit Dialog	Special Cases
Integer	YES		Active Child and Serial Baud Rate Properties have edit dialog boxes.
Boolean	YES		Double-clicking toggles value.
String	YES		Serial Port has an edit dialog box.
Double	YES		
Long	YES		
Float	YES		
Vect 2D	YES		
Vect 3D	YES	YES	
Quaternion		YES	
Filename		YES	
Material		YES	
Resource Entry		YES	
RGB		YES	
List		YES	
Object		YES	
LOD Ranges		YES	
Constraints		YES	
Event		YES	

10

Windows, Viewports, and Viewpoints

This chapter discusses windows, viewports, viewpoints, and their relationships.

Windows and Viewpoints

Each window has a viewpoint associated with it. The viewpoint defines the position and orientation from which the graphical universe is projected to the computer screen and rendered within a window. The viewpoint represents the point of view of the observer. As you navigate through a simulation, you are constantly changing the viewpoint.

The two types of windows are:

- *Application window* – The window in which the simulation displays when you run the simulation as an application within WorldUp, or when your end-users run the simulation using one of the WorldUp players. You can add navigation control panels to application windows, providing a graphical interface from which the end-user can navigate through your simulation.
- *Development window* – The window in the development environment where you can navigate and manipulate the scene as you develop the simulation. For more information on the Development window, see Chapter 8,

Development Window – Navigation and Manipulation. You can set up multiple viewports for the Development window using the Viewport Configuration dialog box. For more information, see "Setting Multiple Viewports" on page 84.

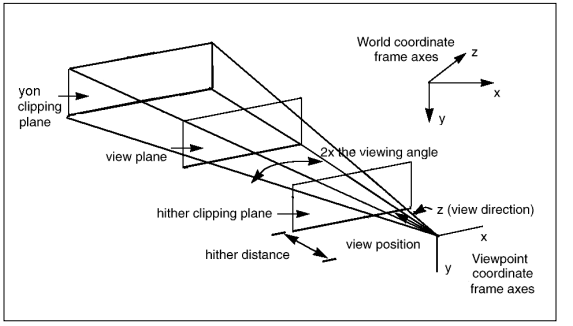
You can use a sensor (such as a mouse) or a recorded path to control a viewpoint. As you move the sensor or play the path, the viewpoint moves automatically enabling you to navigate within a graphical simulation. Using sensors or paths to navigate gives you control over what you see and when you see it.

Each window and viewpoint that you create has a corresponding object in the Type Workview. Viewpoints are created from the Viewpoint object type, Application windows are created from the Window object type, and Development windows are created from the DevWindow object type (which is a subtype of Window). By default, WorldUp provides you with one viewpoint (Viewpoint-1), one application window (Window-1), and one development window (DevWindow-1).

Clipping Planes

For any window, it is important to be aware of the clipping planes that are specified for that window. The Hither Clipping Plane indicates the physical range in front of the viewpoint, before which objects are not rendered in that window. The Yon Clipping Plane indicates the physical range in front of the viewpoint, beyond which objects are not rendered in that window. That is, objects are rendered only in the area between the Hither Clipping Plane and the Yon Clipping Plane.

Clipping planes are controlled by the Hither Clipping and Yon Clipping properties on all Window and DevWindow objects. Search the online Help for more information on these properties.



View Volumes

Creating Viewpoints

In WorldUp, you can create a viewpoint, link the Viewpoint object to the sensor or path that you want to use to navigate the viewpoint, and then associate the Viewpoint object with a window.

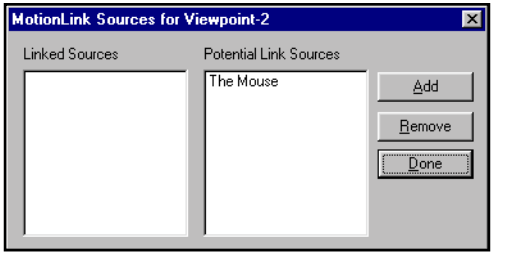
To create a viewpoint

1 Select the Viewpoint object type in the Type Workview.

2 Click the Instantiate Selected Type button.

3 On the Object menu, select Edit MotionLink Sources.

The Motion Link Sources dialog box displays.



4 In the Potential Link Sources box, double-click the sensor or path to which you want to link the viewpoint.

The selected object moves to the Linked Sources box.

5 Click Done.

This creates a motion link between the selected source and the viewpoint so that you can use the source to navigate within any window that uses this viewpoint.

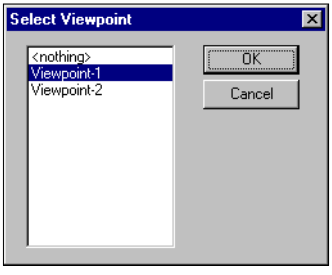
For more information on motion links, see "Motion Links" on page 144.

In the Type Workview under the MotionLink object type, a new object has been created for the link that you just specified. The Source property for the MotionLink object is set to the name of the sensor or path that you specified and the Target property is set to the viewpoint from which you accessed the Motion Link Sources dialog box.

6 In the Type Workview, select the object representing the Application window (under the Window object type) or the Development window (under the DevWindow object type) that you want to associate with the viewpoint.

7 In the Editable tab of the Property pane, double-click the Viewpoint property.

The Select Viewpoint dialog box appears.



8 Click the Viewpoint object that you want to associate with the window, and then click OK.

In the associated window, you can now modify the viewpoint's position and orientation using the sensor or path that you linked to it.

If your motion link's source is a sensor, use the navigation techniques described on "Changing Views" on page 83. If your motion link's source is a path, see "Recording Paths" on page 141 for instructions on playing paths.

Creating a Window

You create a window by creating an object from the Window object type.

To create a window

1 Click the Window object type in the Type Workview.

2 Click the Instantiate Selected Type button.

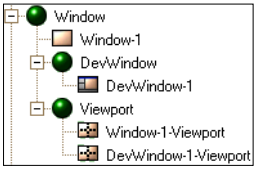
The Select Viewpoint dialog box displays.

3 Click the Viewpoint object that you want to associate with the window, and then click OK.

Note Although you can create multiple Development windows, this practice is discouraged. Instead, use multiple viewports in a single Development window.

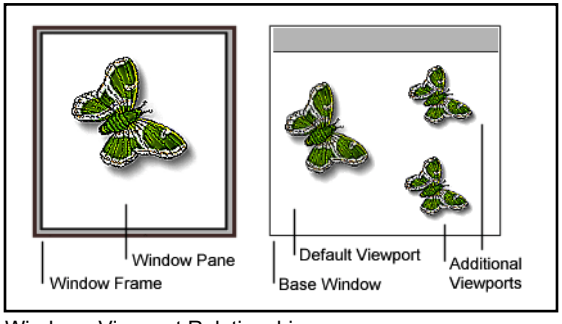
Viewports

Every WorldUp window object contains, by default, a single viewport which covers the entire area of the window and in which the scene is rendered. This viewport is referred to as the Window object's *default* viewport. For each Window object you create in your simulation, WorldUp automatically manages the creation, naming, and deletion of this default viewport.



Viewport Hierarchy

In order to better understand the relationship between windows and viewports, it's often helpful to think of the window as a frame and the viewport as a pane upon which the scene is actually drawn. The window frame (also known as a viewport's base window) can hold many viewports. Likewise, each viewport pane must have a frame to hold it. Unlike traditional window panes, however, viewport panes can be overlapped. In addition, each viewport has it's own viewpoint. This flexibility gives you the power to combine different rendering surfaces for unique situations.



Window - Viewport Relationship

For convenience and backward compatibility, all WorldUp windows come with one default viewport pane whose properties are accessed through it's base window. Additional viewports can be added to any WorldUp window (up to a maximum of eight). Since a viewport has it's own viewpoint, this allows you the flexibility of having multiple views of one or more scenes rendered inside a single WorldUp window.

There are two performance advantages to creating and using multiple viewports in a single window instead of creating and using multiple windows. The first advantage is that performance is improved when using multiple viewports in a WorldUp window instead of using multiple (single viewport) WorldUp windows. The reason for this is that the rendering buffers are cleared and swapped only once for the single window, rather than having to clear and swap for several windows. The second advantage is that the rendering of each viewport is frame synchronized, that is all viewports are rendered on the screen at the same time in a given frame. In contrast, using multiple windows means that WorldUp must process and render the geometry associated with the first WorldUp window before it can process and render the geometry associated with succeeding WorldUp windows. If your application's frame rate is low, there will be a discernible time lag between the updates of each window within one frame.

In addition to performance advantages, the previously mentioned configuration flexibility allows you to create unique solutions to rendering challenges as well as special effects. For example, you can create a rear-view mirror effect by using multiple viewports in a window. Simply add an additional viewport pane to your existing window. Center its position where you would like the rear view mirror to be. Now assign it a unique viewpoint that is looking in the opposite direction of the base window's viewpoint.

Creating Viewports

You create a viewport by creating an object from the Viewport object type and associating it with a window and viewpoint.

To create and configure viewports for the Development window, see "Setting Multiple Viewports" on page 84.

To create a viewport

1 Click the Viewport object type in the Type Workview.

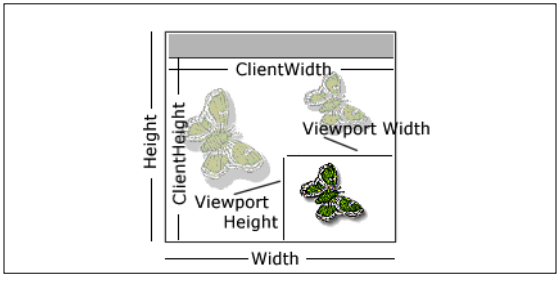
2 Click the Instantiate Selected Type button.

3 Click the Window object that you want to associate with the viewport, and then click OK.

4 Click the Viewpoint object that you want to associate with the viewport, and then click OK.

Sizing Viewports

To change a viewport's size and position, it is important to understand how viewports are embedded into base windows.



Window Size Properties

As with windows, the position of a viewport is specified by the position of its top left corner. The viewport's position is, however, relative to the window, and is represented as *x* and *y* offsets from the top left corner of the base window's client area. The client area represents the drawing area of the window and does not include the window's borders. For example, if a window is positioned at 50,50 in screen coordinates, the default viewport's position is 0,0 since it is relative to the top left corner of the window.

A window object has Client Width and Client Height properties, which represent the size of the window without the borders. As an example, a typical default viewport has the same size as it's base window's Client size properties.

Finally, it is important to note that when a window is resized (either by changing it's properties, or by dragging its frame), all viewports belonging to that window will be scaled accordingly. The default viewport remains the exact size of the client area, while any additional viewports preserve their relative position within the window. You should keep this in mind when sizing your viewports.

The following table gives properties that are specific to viewport size and dimension.

Property	Description	Acceptable Values
Viewport Height	Height of the viewport (pixels)	Unsigned Integer
Viewport Width	Width of the Viewport (pixels)	Unsigned Integer
Viewport XOffset	Location of left side of viewport relative to base window's client area (pixels)	Integer
Viewport Yoffset	Location of top side of viewport relative to base window client area (pixels)	Integer

Since the Viewport object type is a subtype of the Window type, it inherits all of the window properties. These properties all apply in the same manner as they do to a window, with the following exceptions:

- Background Color – All viewports share the same background color as their base window.
- Draw Task (Draw Task 3D) – Each viewport can have its own draw routine. The default viewport's draw routine, however, is associated with the window object's draw task, not the default viewport's draw task.
- Base Window – All viewports have a base window, which is the window object they are embedded in. This property is read-only, since you cannot change a viewport's base window once it is created.

Finally, be aware that the addition of viewports emphasizes the need for explicit window size control. The three window sizes for any given rendering window are client (which is a read-only property since it is computed), window, and viewport. Since viewports are a separate rendering space as defined above, they naturally have differing parameters for size since they do not include the MFC window border. Taking the time to understand the different size properties – Window, Client, and Viewport – can greatly reduce the chance of errors arising from sizing confusion.

Stereo Viewing

WorldUp supports numerous methods for achieving stereoscopic display of your simulation. Before creating a stereoscopic display, consider the specific needs of your simulation, such as whether it will be projected or in an HMD.

Innumerable hardware devices display some form of stereoscopic output, each having advantages and disadvantages. You should discuss these advantages and disadvantages within the context of your simulation environment with the hardware vendor before purchasing your hardware. You should also check `www.sense8.com/support` for an up-to-date list of WorldUp supported devices.

The three most common methods of achieving stereo displays that WorldUp supports are:

- Dual channel – Also referred to as separate channel or multi-channel, this form of stereo uses two separate video signals and is typically used with HMDs.
- Line Interleaved – Also referred to as interlaced or passive stereo, this form uses polarized glasses to isolate left and right eye scan lines.
- Quad Buffered – Also referred to as field sequential or Crystal Eyes, this form uses shuttering glasses to sequentially produce alternating left and right eye images at a rate higher than the human brain is able to distinguish.

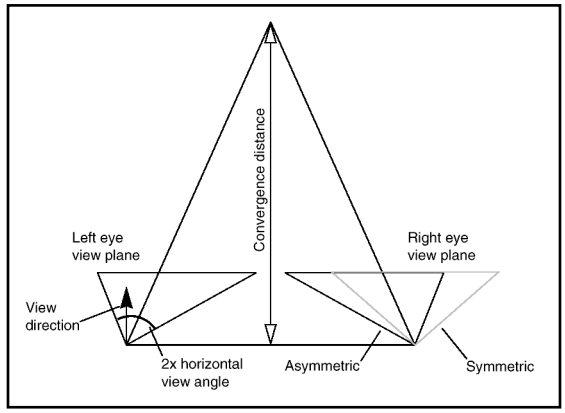
Dual Channel Stereo

WorldUp's support for dual channel stereo employs viewports. Users create a dual channel stereo window by setting the DualChannelStereo property of the Window type to True and instantiating a Window object of that type (similar to other stereo window creation methods). Upon instantiation, WorldUp creates a Window-1 object of size 1280 x 480 (assumes the common 640 x 480 resolution of most HMDs), a default viewport for the Window-1 object, and a secondary viewport for the Window-1 object.

The default viewport is considered the left eye and assumes a dimension of 640 x 480. The secondary viewport is considered the right eye and assumes a 640 x 480 resolution as well. The right eye is also positioned in the right half of the base window.

The borderless parameter must be set to the desired value on the Window type before you create it. Other stereo parameters, such as asymmetric view volume, parallax, convergence, and convergence distance, can be modified either before or after instantiation.

For more information on the View Volume, refer to the figure on page 96. For information on parallax, convergence, and convergence distance, refer to the figure below. For more information on the properties of windows, viewpoints, and viewports, refer to the online Help.



Stereo Parameters

Line Interleaved Stereo

Line interleaved stereo has two modes: software interlaced and hardware interlaced. WorldUp supports both approaches. Both of these approaches draw interleaved left eye/right eye scan lines within a window. All the even lines are left eye and all the odd are right eye (or vice versa). This method of stereo is typically used in conjunction with polarized glasses. One significant drawback to this approach is that your vertical resolution is cut in half.

To use software interlaced mode with WorldUp, you should ensure your video board supports stencil planes and is configured to use them. Once you have configured your video board, you create an interleaved window by setting the Interleaved property of the Window type to True.

Hardware interlaced mode does not require the use of stencil planes, but does require special hardware support. This is an uncommon form of stereo and is typically only supported by Intergraph video boards. To create a hardware interlaced window, set both the Interleaved and Stereo properties of the Window type to True and instantiate it.

Quad Buffered

By far the most popular, quad buffered stereo employs shuttering glasses to present alternating left eye/right eye images. Quad buffered stereo typically requires the following video hardware support:

- A video card with stereo support. This means it is able to generate high enough signal generation frequency, typically from 90–120 hertz, and has a VESA standard 3-pin mini DIN connector on the graphics board to synchronize the signal with the shutter glasses (although there are exceptions to this DIN port for synchronization).
- A monitor capable of refreshing at this frequency.
- Shutter glasses with an interface to allow for synchronization with the video board. Crystal Eyes form StereoGraphics is the standard.

To create a quad buffered stereo window, you first need to prepare your system according to the board manufacturer's instructions for quad buffered stereo. This is typically found in the display settings for your video board.

Once your board is set to the correct frequencies and supported resolutions, set the Stereo property of the Window type to True and instantiate the window. If your window displays contain two completely separate images, one on top an one below, this indicates WorldUp was unable to detect hardware supported for quad buffered stereo. Don't forget to set your viewpoint's Parallax property.

Troubleshooting Stereo Problems

Most stereo problems result from configuration errors or unsupported modes in hardware. The first step is to confirm that your video settings for stereo are set up correctly according to the board manufacturer's specifications. If you are still experiencing difficulty, contact a Sense8 Technical Support Representative, who can check your video configuration against a list of known compatible video boards.

11

Adding 3D Objects

This chapter describes how to create geometric shapes and how to import models into your project using the Model Workview.

Geometries

Objects that you create from subtypes of the Geometry type are graphical objects. That is, they are objects that are visible in the Development window. You cannot create objects or object types directly from the Geometry object type. It exists to provide its subtypes with the necessary properties and to maintain the coherency of the Type pane's structure.

The direct subtypes of the Geometry object type are:

- *Block* – Block objects contain Height, Depth, and Width properties, allowing you to create geometries with a rectangular shape.
- *Cone* – Cone objects contain Initial Radius and Height properties, allowing you to create geometries with a conical shape.
- *Cylinder* – Cylinder objects contain Initial Radius and Height properties, allowing you to create geometries with a cylindrical shape.
- *Imported* – Imported objects contain a Filename property, allowing you to reference geometries that were created with a modeling program.

- *Sphere* – Sphere objects contain an Initial Radius property, allowing you to create geometries with a spherical shape.
- *Text3d* – Text3d objects contain Font File and Text String properties, allowing you to add 3D text to your simulation. Color and texture information are contained in the 3D font file that you specify.

  For information on creating your own 3D font, see page 195.

Creating Primitives

Primitives are three-dimensional basic geometric forms stored as a collection of polygons. WorldUp supplies four object types from which you can create primitives: Block, Cone, Cylinder, and Sphere. You can edit or combine primitives with other objects to make more complex objects.

To create a block, cone, cylinder, or sphere

1 In the Nodes pane of the Scene Workview, select the Block, Cone, Cylinder, or Sphere object type located under Geometry.

2 Click the Instantiate Selected Type button.

  The new object appears in both the Nodes pane and the Scene Graph pane.

  If you cannot see the new object in the Development window, see "Viewing Graphical Objects" on page 193 for possible solutions.

  By default, all graphical objects are positioned at the center of the universe upon creation. If you created multiple graphical objects, only the largest object is visible until you translate some of the smaller objects to a new location or move the viewpoint inside of the largest object.

3 In the Scene Graph pane, select the new object.

4 Select the Important tab on the Property pane to modify the dimensions, material (color), and number of polygons for the primitive, .

  For information on modifying property values, see "Modifying a Property Value" on page 91.

5 Modify the Important properties to achieve the results that you want.

  For a description of the properties that are specific to each primitive, search on Block (Geometry Subtype), Cone (Geometry Subtype), Cylinder (Geometry Subtype), or Sphere (Geometry Subtype) in the online Help.

Creating 3D Text

You can create three-dimensional text using the Scene Workview.

To create 3D text

1 In the Nodes pane, click the Text3d object type, located under Geometry.

2 Click the Instantiate Selected Type button.

  The new object appears in both the Nodes pane and the Scene Graph pane.

  Note The Font File property for the Text3d object type indicates the name of the font file that will be used to create the object. If the directory path in which this file is located does not exist in WorldUp's search list, the object will not be created and an error message will display in the status window.

  For instructions on how to add directory paths to the search list, see "Configuring Directory Paths" on page 39. (The default value for the Font File property is RCFONT3D.NFF, which is located in the Models directory of the directory in which

you installed WorldUp. By default, this Models directory should already be on your search list unless you manually removed it.)

If you cannot see the new object in the Development window, see "Viewing Graphical Objects" on page 193 for possible solutions.

3 In the Scene Graph pane, click the new object.

4 In the Property pane, click the Important tab.

5 Select the Text String property and slowly click on it again to get the text edit box.

6 Type the text that you want the object to display in the Development window.

By default, the Text String is “Sample String!”

For a description of the remaining properties that are specific to Text3d objects, search on “Text3d (Geometry Subtype)” in the on-line help.

Note You can create your own 3D font and point to that font using the Filename property. See "Fonts" on page 195 for more information.

Importing Models from Third Parties

You can import models into your project that were created using a third party modeling program.

To import a model to WorldUp

1 In the project window, select the Model Workview

2 Click Import New Model.

The Open dialog box displays.

3 Navigate to the directory containing the file that you want to import.

4 Double-click the file name.

The Import Model Parameters dialog box displays.

5 Make any necessary modifications to the import model parameters, which are described in the following table:

Scale for Stretch and Position	This is the amount by which a geometry is stretched and positioned. This value applies to all geometries in the model file. This option is useful for scaling multi-object models that you intend to import as multiple objects, since each object will maintain its spatial relationship to the model as a whole. For single-object models or multi-object models that have been combined into a single node, modifying this setting is no different than modifying the object’s Stretch property.
Combine All Geometry into Single Node	This option affects multi-object models. When this option is checked, the objects that make up a model are merged into a single object. In the Model Workview, the model will contain a single entry, called <All Geometry>. When this option is unchecked, each object in a multi-object file remains distinct. In the Model Workview, the model will contain a separate entry for each object, using the names specified in the model file for each object.
Treat Geometry Midpoint as Object Center Point	In a 3D Studio file, there is one origin and a geometry is constructed relative to that. In WorldUp, each geometry has its own origin, and the geometry has a position in space. With this option enabled, 3DS objects are given a reasonable origin and the correct position in space.

6 When you are finished setting the import model parameters, click OK.

The newly loaded model is added to the Imported Models pane and displayed in the Preview pane. Its properties are shown to the right of the Preview pane. In the Imported Models pane, all of the geometries available in the file are displayed under the Root node which has a name the same as the file name. Geometries that appear in the Imported Models pane are also known as model entries.

In WorldUp, all Imported objects contain Filename and Entry properties, indicating the name of the model file and the specific entry within that model file that the object references.

To add an object from an Imported Model to your scene

1 In the Imported Models pane of Model Workview, expand the file name you want the object to import from.

2 Click on the object you want to add to your scene.

The object is displayed in Preview pane with its properties on the right.

3 Drag and drop the object to the desired location in Scene Graph pane.

You can rearrange the scene graph later if you didn't drop the object in the right place.

If you cannot see the new object in the Development window, see "Viewing Graphical Objects" on page 193 for possible solutions.

To add all available objects from a loaded model to your scene

1 In the Imported Models pane, click on the file name of the model you want to add to the scene.

2 Drag and drop the object to the desired location in the Scene Graph pane.

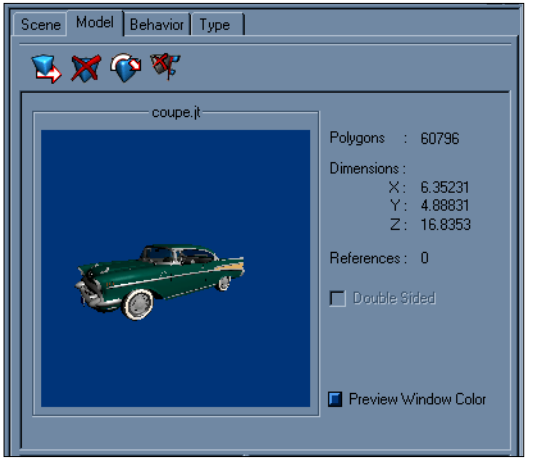
If you cannot see new objects in the Development window, see "Viewing Graphical Objects" on page 193 for possible solutions.

Using the Model Workview

This section describes various tools available in Model Workview for importing, previewing, editing, and adding models to your scene.

Previewing Geometries

When an object is selected in the Imported Models pane, the Preview pane shows the object and all of its children.

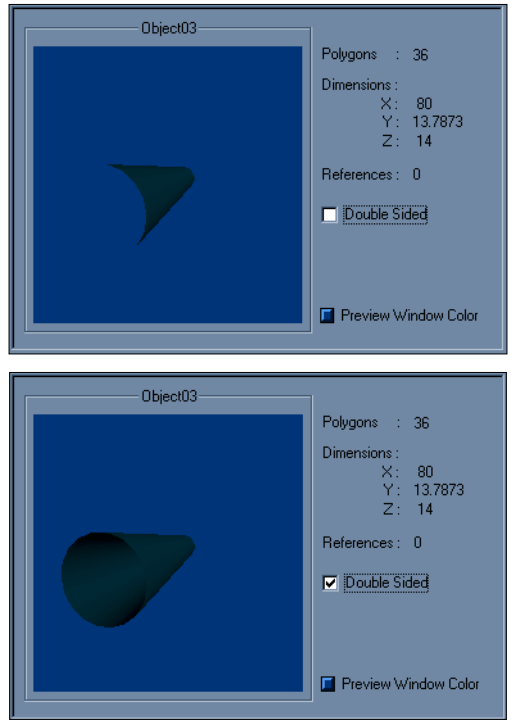


A Car Model Loaded in Model Work view

The Preview title shows the model's file name that is selected in the Imported Models pane which is the root of all objects contained in the file. If a model entry is selected in the tree, the title displays the name of that object.

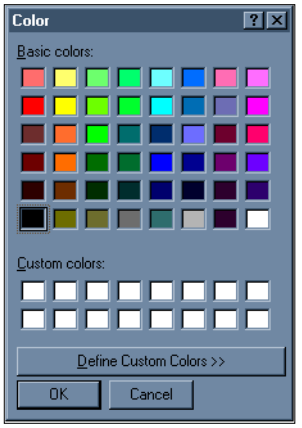
To the right of the preview, the properties relevant to the object are displayed. First is Polygons which shows the combined polygon count of the object and its children. Dimensions shows the combined size of the object and its children in *x*, *y*, and *z* directions. References represents the number of times the object is used in the scene graph. The number of references of a model or model entry is also shown in parenthesis after the model or model entry name in the Imported Models pane.

You can make an object double-sided or single-sided by checking the Double Sided check box. The first feature is useful if a geometry is imported single-sided and you want to see the geometry from both sides as shown in the figure below. You would want to make a geometry single sided if the geometry is not going to be viewed from the other side. Making a geometry single-sided increases the performance during rendering.



Making a Geometry Visible from Both Sides

The color in the Preview window can be customized by clicking on the Preview Window Color button and selecting a color of your choice from the color dialog. If the model has a dark color, you might want to set the window color to a lighter one.



Customize the Preview Window Color

The model can be viewed from different angles by clicking and dragging with the mouse in the Preview window

Reloading a Model

You can reload a model already loaded into the Model Workview or added to the scene.

To reload a model

1 Select the model in the Imported Models pane.

2 Click the Reload Model button.

This option allows you to modify the model file in a modeling program and see the changes in WorldUp without reloading the universe. This option can also be available by right-clicking on a model in the Imported Models pane.

Removing Imported Models

Before you can remove a model from the Model Workview, any references to the selected model or model entry must be removed.

To unreference a model

➤ Click the Unreference Model button in the toolbar of the Model Workview.

You can also call Unreference Model by right-clicking on Model Filename in the Imported Models pane. Unreference Model deletes all objects from the scene that refer to the selected model. The model remains in the Model Workview.

To remove a model from the Model Workview

1 In the Model Workview, select the model you want to remove.

2 Click on the Remove Model from Project and Memory button in the toolbar.

You also can right-click on a model and select Remove Model.

Note All unused imported models are automatically removed when you close the project.

Re-using Imported Geometries

When you re-use an imported geometry, you allow multiple objects to share the same model entry. Thus, if you scale, adjust the pivot point, or optimize an imported geometry, you are modifying the model entry itself, and those settings will also be reflected in any object that shares that model entry. Any other modifications that you make to the object, such as translation and rotation, do not affect the model entry and remain independent.

To re-use an imported model

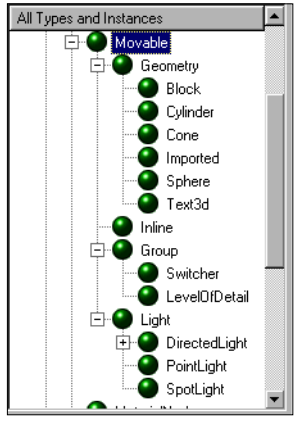
➤ Drag the object from the Imported Models pane into the Scene Graph pane.

You can repeat this procedure to reuse the model as many times as you want.

12

Editing 3D Objects

WorldUp provides various ways of changing the appearance of an object. These tools allow you to translate, rotate, and scale objects and change their Material properties. The objects that can be translated and rotated are known as *Movables*. These are the objects whose type is derived from the Movable object type. They are Geometry, Group, and Light as shown below in the Type Workview.



Movable Objects as Displayed in the Type Workview

You cannot create objects or object types directly from the Movable object type. It exists to provide its subtypes with the necessary properties and to maintain the coherency of the Nodes pane structure.

The direct subtypes of the Movable object type are:

- *Geometry* – A Geometry object (also called a *graphical object*) is an object that you can see in the Development window. See "Geometries" on page 103 for a detailed description of Geometries.
- *Group* – A Group object acts as a container to allow you to manipulate multiple nodes at once. See "Groups" on page 25 for a detailed description of Groups.
- *Light* – Light objects add various lighting effects to your geometries. See Chapter 13, *Lights* for a detailed description of Lights.

Translating and Rotating Movables

The three ways that you can translate (move) and rotate Movable objects are:

- drag the object with the mouse in the Development window
- use the Property pane to modify the object's Translation and Rotation properties
- use the Position Object dialog box

These methods are described in the following sections. The last two methods can also be used to scale graphical objects.

Dragging Objects in the Development Window

Because Lights and Groups cannot be seen in the Development window, you use different methods for dragging each of the following types of Movable objects:

- Graphical objects (Geometries)
- Non-graphical objects (Lights or Groups)

To translate or rotate graphical objects

1 Select the Translate Object or the Rotate Object button in Development window toolbar.

For more information on various Development window toolbar buttons, see Chapter 8, *Development Window – Navigation and Manipulation*.

2 Select the object in the Development window and drag the mouse to translate/rotate the object.

If you have trouble dragging a particular object, see "Locking a Selected Object" on page 111 for help.

Note You must click on the object itself. If you click outside the object's bounding box, or on an empty area within the object's bounding box, the object will become deselected.

3 Release the mouse button to stop translating/rotating the object.

Note You can revert to the object's original position by selecting Undo Last Move on the Edit menu immediately after translating or rotating the object.

To drag Lights and Groups

1 Select the Translate Object or the Rotate Object button in Development window toolbar.

2 In the Scene Graph pane, click the Light and/or Group objects that you want to translate/rotate.

Since the selected objects cannot be seen in the Development window, you need to lock on to the object to provide a visual target that you can drag.

3 If it is not already depressed, click the Lock Selected button on the Development window toolbar.

4 To translate/rotate the selected object, click the left mouse button in Development Window and drag.

5 Release the mouse button to stop translation/rotation.

Since the translation/rotation value of a child object is always in relation to its parent's coordinate frame, child objects move in the Development window to maintain their positions within the changing coordinate frame. The translation/rotation values for child objects remain unchanged.

Note You can revert to the object's original position by selecting Undo Last Move from the Edit menu immediately after translating or rotating the object.

Locking a Selected Object

You can lock the view and the selection to a particular Movable object in the Development window. This option is necessary if you want to drag non-graphical objects. Also, this feature is useful when you want to modify a graphical object in your simulation that is hard to grab (for example, it may be very thin, or obstructed by another graphical object).

To lock a Movable object

1 Select the object in the Development Window or, if the object is non-graphical. in the Scene Graph pane of the Scene Workview.

2 Click on the Lock button in the Development window toolbar.

When an object is locked, no other object can be selected or dragged in the Development window.

Note You can select a new object in Scene Graph pane even if Lock button is depressed.

Translating and Rotating Using the Property Pane

This section describes how to modify the Translation and Rotation property values for an object from the Property pane.

Translating Movables from the Property Pane

You can translate Movable objects from the Property pane by modifying the Translation property value for the object.

The Translation property has a Vect3d data type. You can enter specific coordinates in the Property pane's text box, or you can use the Translation dialog box to visually place the objects.

For more information on WorldUp data types, see the *WorldUp Programmer's Guide*.

To enter specific coordinates for the Translation property

1 Select the Movable object that you want to move.

2 In the Property pane, click the Editable tab.

3 Slowly double-click in the Value column of the Translation property.

4 Enter the coordinates (*x, y, z*) for the new location.

5 Press ENTER.

The selected objects move to the new location in relation to their parents' coordinate systems. Any selected objects that have the same parent will move to the same global location.

To use the Translation dialog box

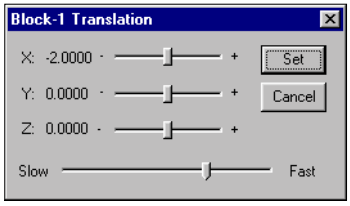
1 Select the Movable object that you want to move.

Note Select only one object. You cannot access the Translation dialog box with multiple objects selected.

2 In the Property pane, click the Editable tab.

3 Double-click the Translation property.

The Translation dialog box displays.



4 Drag the X, Y, and Z sliders as appropriate to position the object.

You can see the object move in the Development window as you drag the sliders.

5 To adjust the speed at which the object moves, drag the slider at the bottom toward Slow or Fast.

6 When you are finished, click Set to apply the new value and close the dialog box.

Rotating Movables from the Property Pane

You can rotate Movable objects from the Property pane by modifying the Rotation property value for the object.

The Rotation property has an Orientation data type, which is a quaternion. This data type is used for accuracy and efficiency since, unlike Euler angles, there is only one way to achieve a result with a quaternion.

For more information on WorldUp data types, see the *WorldUp Programmer's Guide*.

However, unlike the Vect3d data type used with the Translation property, you are not expected to calculate an intelligent value to use for the Orientation value. Instead, WorldUp provides the Rotation dialog box, which uses Euler angles rather than quaternions.

Euler angles provide an easy way to achieve the rotation that you want, but they are inefficient since there are multiple ways to achieve that rotation. For example, (0, 180, 0) and (0, -180, 0) result in the same position. When you use the Rotation dialog box, WorldUp converts the angles that you set into a quaternion and the Rotation property is updated with the converted value.

To use the Rotation dialog box

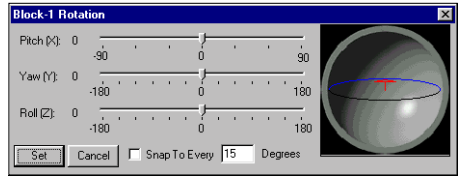
1 Select the Movable object that you want to rotate.

Note Select only one object. You cannot access the Rotation dialog box with multiple objects selected.

2 In the Property pane, click the Editable tab.

3 Double-click the Rotation property.

The Rotation dialog box displays.



4 Drag the X, Y, and Z sliders as appropriate to position the object where you want it.

You can see the object rotate in the Development window as you drag the sliders.

Alternatively, you can click inside the sphere at the right of the Rotation dialog box and drag the red arrow to achieve the desired rotation. (Use the right mouse button to adjust the viewpoint of the sphere.)

5 To cause the rotation of an object to snap at a specified interval, check the Snap To Every box at the bottom of the dialog box and type the number of degrees to which you want the object to snap.

6 When you are finished, click Set to apply the new value and close the dialog box.

The Euler angles that you specified are converted to a quaternion and the Rotation Property value is updated.

Translating and Rotating Using the Position Object Dialog Box

The Position Object dialog box provides a convenient way to rotate, translate, and scale objects all from one dialog box. All Movable objects can be translated and rotated, but only graphical objects can be scaled. For information on scaling geometries, see "Scaling Geometries" on page 114.

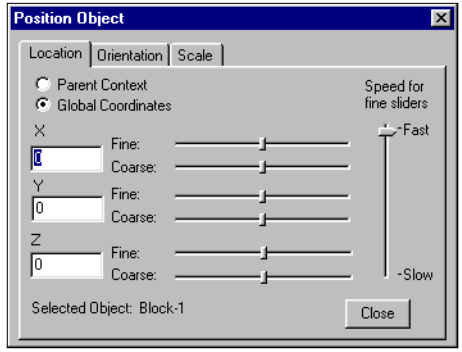
To translate Movables from the Position Object dialog box

1 In the Scene Graph pane of the Scene Workview, select the Movable object that you want to move.

Note Select only one object. You cannot translate multiple objects at once from the Position Object dialog box.

2 Right-click and select Position Object.

The Position Object dialog box displays.



3 Click the Location tab.

4 At the top of the tab, click the coordinate system in which you want to move the object.

- *Parent Context* – The coordinate system of the object's parent in the scene graph.
- *Global Coordinates* – The world coordinate system.

For more information on coordinate systems, see "Coordinate Systems" on page 13.

5 Type specific coordinates for each axis, or use the Fine and Course sliders to move the object.

- *Fine* – Increments the value up or down by moving the slider to the left or right of the center. When you release the mouse, the slider handle returns to the center.
- *Course* – Moves an object a certain percentage of the universe's current length in that axis. For example, if you click at the right-most edge of the slider for the x-axis, the selected object moves to the farthest point possible along the positive direction of the x-axis in the current extents of the universe.

6 To adjust the speed at which the object moves, drag the slider at the right toward Slow or Fast.

7 When you are finished, click Close.

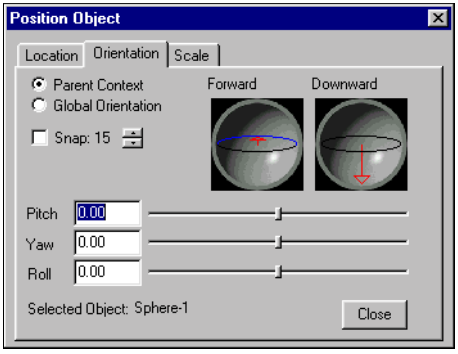
To rotate Movables from the Position Object dialog box

1 In the Scene Graph pane of the Scene Workview, select the Movable object that you want to rotate.

Note Select only one object. You cannot rotate multiple objects at once from the Position Object dialog box.

2 Right-click and select Position Object.

3 The Position Object dialog box displays.



4 To rotate a Movable object, click the Orientation tab.

5 Click the coordinate system in which you want to rotate the object.

- *Parent Context* – the coordinate system of the object's parent in the scene graph.
- *Global Orientation* – the world coordinate system.

For more information on coordinate systems, see "Coordinate Systems" on page 13.

6 Rotate the object using one of the following methods:

- Type specific Euler angles for each axis. *Pitch* refers to the x-axis, *Yaw* refers to the y-axis, and *Roll* refers to the z-axis.
- Drag the Pitch, Yaw, and Roll sliders as appropriate to position the object.
- Click inside either sphere at the top-right of the Position Object dialog box and drag the red arrow to achieve the desired rotation. Use the right mouse button to adjust the viewpoint of the sphere.

7 To cause the rotation of an object to snap at a specified interval, check the Snap box and use the up and down arrows to specify the number of degrees to which you want the object to snap.

8 When you are finished, click Close.

The Euler angles that you specified are converted to a quaternion and the Rotation property value is updated.

Scaling Geometries

Scaling a geometry involves specifying along which axis you want to scale the object and by what factor.

For example, suppose you have a cube with a height, depth, and width of 3. If you stretch the cube by a factor of 2 along the x-axis, and 1 along the y and z-axes, the cube is now rendered 6 units in width, and 3 units in height and depth. However, the Height, Depth, and Width property values all remain set to 3. The scale factors are stored in the object's Stretch property.

Note If you scale a geometry that is also being used elsewhere, all other objects that also reference that resource entry will be scaled in the same manner.

The two ways that you can scale geometries are:

- use the Property pane to modify the object's Stretch property
- use the Position Object dialog box

Note Information on translating and rotating geometries is described in "Translating and Rotating Movables" on page 110, since all Movable objects can be translated and rotated, but only geometries can be scaled.

Scaling Geometries from the Property Pane

You can scale graphical objects from the Property pane by modifying the Stretch property value for the object. Objects are scaled within their local coordinate frame only.

The Stretch property has a Vect3d data type. You can enter specific scale factors in the Property pane's text box, or you can use the Stretch dialog box to visually scale the objects.

For more information on WorldUp data types, see the *WorldUp Programmer's Guide*.

Note Visually scaling the objects using the Stretch dialog box is most useful if you do not know the exact factor by which you want to scale the object, and it is not important that the object be scaled uniformly along each axis.

To enter specific scale factors for the Stretch property

1 Select the Geometry object that you want to scale.

2 In the Property pane, click the Editable tab.

3 Single-click the Stretch property.

4 Click the Stretch property again to open the text edit box and enter the scale factors for each axis (x, y, z).

5 Press ENTER.

The selected objects stretch accordingly in the Development window.

To use the Stretch dialog box

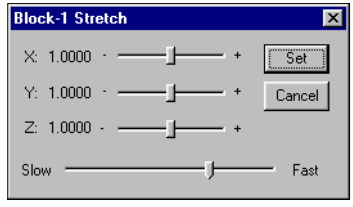
1 Select the Geometry object that you want to stretch.

Note Select only one object. You cannot access the Stretch dialog box with multiple objects selected.

2 In the Property pane, click the Editable tab.

3 Double-click the Stretch property.

The Stretch dialog box displays.



4 Drag the X, Y, and Z sliders as appropriate to stretch the object along any axis.

You can see the object stretch in the Development window as you drag the sliders.

5 To adjust the speed at which the object moves, drag the slider at the bottom toward Slow or Fast.

6 When you are finished, click Set to apply the new value and close the dialog box.

Scaling Geometries from the Position Object Dialog Box

You can also scale geometries using the Position Object dialog box.

To scale geometries from the Position Object dialog box

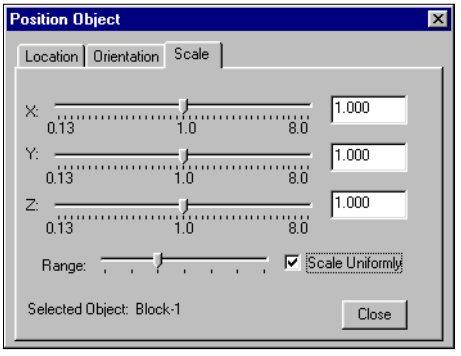
1 In Scene Graph pane of Scene Workview, select the Geometry object that you want to scale.

Note Select only one object. You cannot scale multiple objects at once from the Position Object dialog box.

2 Right-click and select Position Object.

For information on using the Position Object dialog box to translate and rotate Movable objects, see "Translating and Rotating Using the Position Object Dialog Box" on page 113.

The Position Object dialog box displays.



3 To scale a geometry, click the Scale tab.

4 If you want to scale the object by the same factor along each axis, check the Scale Uniformly box.

5 If you have checked the Scale Uniformly box, drag on any one of the axis sliders and all three move together. Or, type a specific scale factor in one of the three text boxes and the other two values automatically adjust at the correct ratio.

6 If you have unchecked the Scale Uniformly box, drag the slider or type a specific scale factor for the axis along which you want to scale the object.

Note Objects are scaled within their local coordinate frame only. You might want to lock the object (click the Lock button on the Development window) to display the axes for the object and see how the reference frame is oriented.

Adjusting a Geometry's Pivot Point

A *pivot point*, also known as an *origin point*, is the point around which an object rotates or from which it stretches. By default, the pivot point of a graphical object created in WorldUp is located at the center of the object. However, you can change the location of a pivot point.

The location of an object's pivot point (or, its distance from the geometry's center) is stored in the object's Origin Offset property. The Origin Offset property is always in reference to the object's local coordinate frame. Thus, a value of 0, 0, 0 always places the pivot point at the center of the object.

You can move an object's pivot point by dragging it in the Development window, or by modifying the Origin Offset property from the Property pane.

To move an object's pivot point from the Property pane

1 Select the Geometry object whose pivot point you want to move.

2 Click the Lock Selected button if it is not already depressed.

3 In the Property pane, click the Editable tab.

4 To modify the Origin Offset value, do one of the following:

- Double click slowly to open the text edit box, type a new value, and press ENTER.
- If only one object is currently selected, double-click the Origin Offset property. From the Origin Offset dialog box, drag the X, Y, and Z sliders to position the pivot point and click Set.

To drag an object's pivot point in the Development window

1 Select the Geometry objects whose pivot points you want to move.

2 In the Development window, click the Lock button if it is not already selected.

3 Press and hold the SHIFT key and drag with left mouse button to move the pivot point.

To see the effect of the pivot point's new location, rotate the object as described on "Dragging Objects in the Development Window" on page 110.

Using Materials to Change Object's Appearance

A *material* is a combination of light and color attributes that you use to define the appearance of a geometry. Geometries either emit light, reflect light, or both. This light is manifested as color. When designing a geometry, two kinds of color are considered:

- The colors used in the geometry itself.
- The colors of the light playing on the geometry.

A realistic image of a geometry includes many colors – and potentially many ways of reflecting light. You use a separate material to specify each of these differences in appearance.

Each material has the following properties:

Diffuse – The color reflected from the material in direct light.

Ambient – The color reflected from the material in shadow.

Specular – The color reflected from the highlights of the geometry. The Specular material property is what makes a geometry appear to be *shiny* with highlights appearing on its surface. Usually, the specular highlight is white, which means that it reflects the color of the specular light (which is also usually white).

Emissive – The color produced (not reflected) by the material even when there is no light. A geometry with this property can be seen even when no lights are contained in the scene. However, the Emissive light does not illuminate other geometries in the area. This material property is used less often than the others.

Shininess – The narrowness of focus of specular highlights. This has no meaning if the specular color is black. (Lighting of geometries rendered with material properties is an *additive* process; a black specular highlight doesn't darken the geometry, it simply doesn't contribute to a light highlight on the geometry.) The lower the Shininess value, the more

spread out the highlight; the higher the Shininess value, the sharper the highlight. A high value for Shininess makes an object look shiny.

Translucency (Opacity) – The extent to which the color value of a pixel is combined with the color value behind it, giving the effect of a transparent surface.

In WorldUp the two ways you can add materials to an object are by using the Material Node or by using the Material Table.

Applying Material Using the Material Node

Using the Material node is the most flexible method for adding materials in WorldUp. By this method, materials can be created and edited. They can also be saved for re-use in the development environment. Material properties of all types of objects, including imported models, can be changed using material nodes.

To apply material to object using Material Node

1 Drag and drop a Material type onto the Scene Graph pane above the object(s) to which you want to apply the materials.

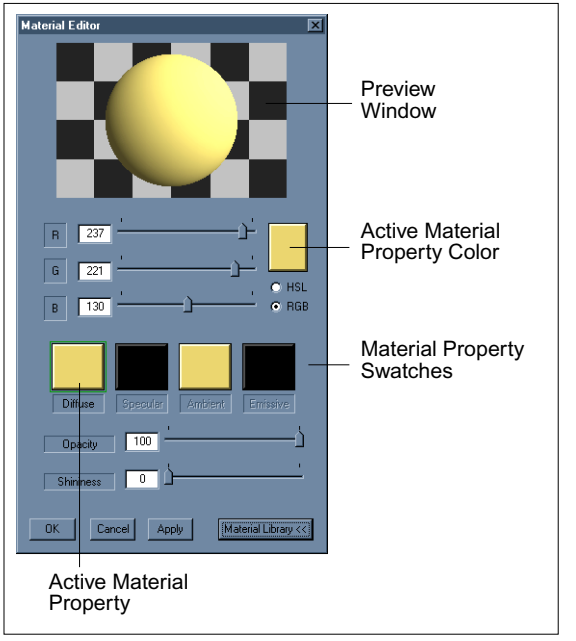
2 Set the UseMaterialNode property of the object(s) to TRUE.

This allows the object to use the Material properties of its parent Material type instead of the default Material Table.

3 Right-click on the Material object you just added to the Scene Graph pane to launch the Material Editor.

The Material Editor allows you to modify or organize materials and has two parts:

- Material Editor – Allows material properties to be previewed and modified.
- Material Library – Allows you to organize, save out, and re-use materials.



Material Editor

The current Material properties applied to a lit sphere are displayed in the Preview window. This allows you to see the changes in Material properties as you edit them. The checker-board background is useful when applying transparency to see the effect.

To set the color values of the Diffuse, Specular, Ambient, or Emissive properties

1 Single click on the desired material property swatch.

2 Enter color values for Red, Green, and Blue directly or use the sliders to select the right value.

You can toggle between RGB and HSL to enter values in RGB or HSL format.

3 Change theOpacity or Shininess properties using sliders or by entering the values directly.

4 Click Apply to see how the updated material looks when applied on the objects.

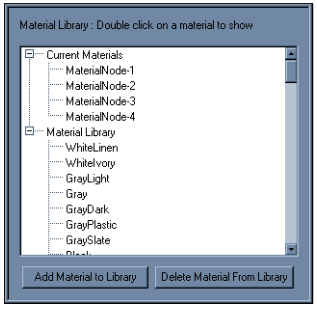
5 Click OK.

Using the Material Library, you can apply already available materials to your objects.

To use the Material Library

1 Click the Material Library button on the Material Editor.

The Material Library dialog box appears.



The tree shows current materials used in WorldUp and materials currently stored in the library.

2 Double-click on a material item to use that as your material and display it in the Preview window.

You can also add and save the material you just created to use later.

To add your material to the library

1 Click the Add Material to Library button.

2 Enter a name for the material.

You can also rename a material in the library by double-clicking on it with a pause.

To save the changes you made

1 Close the Material Library by clicking on the Material Library button in the Material Editor.

You are asked to save the changes.

2 Click OK.

The library is saved as wupMaterialLibrary.mat in your application directory. The file must be writable in order to save the changes.

To delete a material from the library

1 Select a material in the Material Library dialog box.

2 Click the Delete Material From Library button.

3 Save the changes when prompted while closing the Material Library dialog box.

Applying Material Using the Material Table

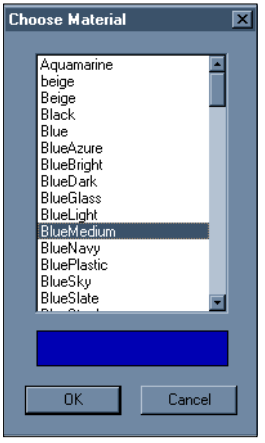
Using the Material Table is the easiest way to add material to an object in WorldUp. Material can only be selected from a given set of materials in the Material Table. Material properties of imported models can not be changed this way.

To apply material to an object this way

1 Select the object in the Scene Graph pane.

2 Select its Material property in the Property pane.

The Choose Material dialog box appears.



3 Select a material from the list and click OK.

Textures

WorldUp preserves texturing of an object as a textured object is imported into WorldUp. However, you must make sure that the path where the texture is stored is added to the path settings.

Textures to WorldUp objects must be applied programmatically, either through WorldUp's scripting language or using the WorldUp Plug-in Kit. Refer to the SetTexture command in the *WorldUp Programmer's Guide* and the *WorldUp Plug-in Author's Guide*.

13

Lights

To understand the effect of lights in WorldUp, you must first understand the concept of materials.

The Effect of Light on Materials

A *material* is a combination of light and color attributes that you use to define the appearance of a geometry or collection of geometries. You can use the default materials supplied with WorldUp, or you can use a modeling application to create, edit, and save customized material information.

Geometries either emit light, reflect light, or both. This light is manifested as color. When designing a geometry, you need to consider two kinds of color:

- The colors used in the geometry itself.
- The colors of the light playing on the geometry.

A realistic image of a geometry includes many colors – and potentially many ways of reflecting light. You use a separate material to specify each of these differences in appearance.

Each material has the following properties:

- *Diffuse* – The color reflected from the material in direct light.
- *Ambient* – The color reflected from the material in shadow.

- *Specular* – The color reflected from the highlights of the geometry. The specular material property is what makes a geometry appear to be "shiny" with highlights appearing on its surface. Usually, the specular highlight is white, which means that it reflects the color of the specular light (which is also usually white).
- *Emissive* – The color produced (not reflected) by the material even when there is no light. A geometry with this property can be seen even when there are no lights in the scene, however, the emissive light does not illuminate other geometry in the area. This material property is used less often than the others.
- *Shininess* – The narrowness of focus of specular highlights. This has no meaning if the specular color is black (lighting of geometry rendered with material properties is an "additive" process; a black specular highlight will not darken the geometry; it simply won't contribute to a light highlight on the geometry). The lower the shininess value, the more "spread out" the highlight; the higher the shininess value, the sharper the highlight. A high value for shininess makes an object look shiny.
- *Translucency(Opacity)* – The extent to which the color value of a pixel is combined with the color value behind it, giving the affect of a transparent surface.

Without light, material properties that are dependent on light, such as Diffuse and Shininess, are ignored when the object is rendered.

Working with Lights

In WorldUp, you can manipulate the lighting of graphical objects in a scene to create many different kinds of effects. Lights can illuminate and enhance the visibility of objects in the universe. Light in a scene can come from several light sources that can be individually turned on and off. Some light comes from a particular direction or position, and some light is a combination of many light sources.

WorldUp can render up to eight lights at a time at any branch of the scene graph. Each light affects only its child nodes and nodes that are at the same hierarchical level and below it in the scene graph. Be sure to design a scene so that none of its objects are lit by more than eight lights.

Different Types of Light

WorldUp uses four types of light – one that exists as a property of the universe (Ambient), and three that exist as objects that you can add to your simulation (Directed, Point, and Spot).

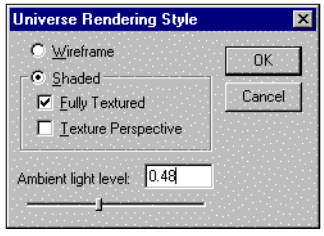
Ambient Light

Ambient light is background light that illuminates all graphical objects equally, regardless of their position or orientation. It comes from no particular source. By default, ambient light is always present in a WorldUp application, even when your simulation contains no light.

To set the level of ambient light

1 Click the Select Render Style button.

The Universe Rendering Style dialog box displays.



2 Drag the slider for Ambient Light Level, or type a value between 0.0 and 1.0 in the text box, with 0.0 being total darkness and 1.0 being total brightness.

3 Click OK.

Note You can also set the ambient light level by modifying the Ambient Light property value for the Universe object.

Directed, Point, and Spot Light

A *directed light* is a light that is located far away but comes from a single direction. Changing the distance between a directed light and an object has no effect because the light is so far away that its rays become parallel by the time they reach the object. Use directed light to give the effect of daylight, making all objects equally visible. By default, WorldUp provides a Directed Light object (Light-1) with each new universe.

A *point light* is a light that comes from a specific position, traveling outward radially in all directions from that position. A light bulb is an example of a point light source. Use point light to make one area appear more illuminated than another.

A *spot light* is a beam of light that illuminates only a small area. Unlike point light, which radiates outward in all directions, spot light produces a cone of illumination. An automobile headlight is an example of a spot light source. Use spot light to center attention on a specific object by displaying the object within a cone of light.

To create a directed, point, or spot light

1 In the Nodes pane, select the DirectedLight, PointLight, or SpotLight subtype under the Light node, and drag the object to the desired location in the Scene Graph pane.

2 If necessary, select the new object and drag it to the correct location in the scene graph so that it illuminates only the appropriate objects.

3 Once the object is created, you may want to modify the object's properties to control factors such as the light's name, color, and location.

Note Information on how to translate and rotate Movable objects, such as Groups, is described in "Translating and Rotating Movables" on page 110.

Lights and Sensors

You can link Sensor objects to lights, or any other Movable object. When you use a Sensor object to control a light, input from the device causes the light to move or be redirected.

For information on linking a Movable object to a sensor, see "Motion Links" on page 144.

Performance Impact of Lights

Since WorldUp needs to perform calculations to determine how much light each Geometry receives from each light source, increasing the number of active lights can adversely affect performance.

In general, spot lights are the most computationally expensive, followed by point lights, and then directed lights. For any simulation you must decide the balance between the simulation's visual quality and it's performance. These two issues typically have an inverse relationship, and it is up to the developer to decide when image quality must be sacrificed to sustain interactive frame rates.

In this section we introduced some of the features WorldUp offers you to improve the realism of your scene. These elements often come at a performance cost. Some, however, cost more than others (often depending on your particular simulation), so we also

discussed ways of optimizing performance without completely compromising your simulation's aesthetics.

Lighting is essential for any simulation as a requisite visual cue for resolving depth, shape, and object relationship. Lights should also be used carefully, however, since each light adds significant computational overhead with respect to shading your scene. It is recommended you keep the number of lights in your scene at or below four, depending on your hardware. The maximum number of lights you may have in your scene is eight.

 For rendering performance, you should always disable light nodes if they are not in use. Setting their intensity to 0 will not speed performance.

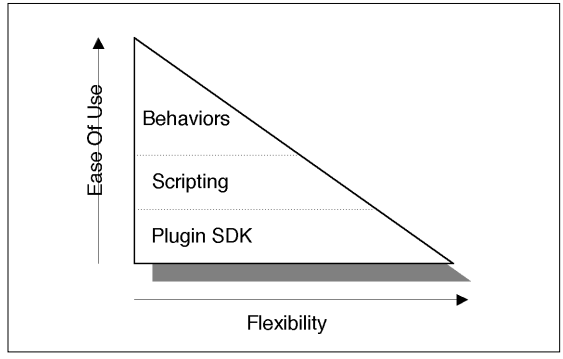
WorldUp provides you with four types of lights: Ambient, Directed, Point, and Spot. Each behave as their name implies. Ambient is the exception, however, as it is a global illumination approximation and as such is controlled as a property of your Universe object. The remaining three are node types in the Type pane.

Note *Remember light nodes are attribute nodes. As such, they only illuminate objects that are visited after and below them during scene graph traversal and exist at their hierarchical level and below.* For an example, see "Organizational Nodes" on page 25.

14

The Behavior System

The WorldUp Behavior System is new to Release 5. It represents the third tier in our 3-tier simulation building model as shown in the figure below.



R5 API Triangle

The purpose of the Behavior System is to provide a truly visual programming paradigm that allows you to rapidly assemble 3D simulations without programming. In addition, the system gradualizes the script learning curve by providing a natural evolution into scripting once your need to customize behaviors increases and your simulation's complexity requires a custom fit. Finally, it serves as a mechanism for producing reusable behavior components that you can build, re-use in other simulations, and share with other users.

In short, the core purpose of the R5 Behavior System is threefold:

- Provide a mechanism for visually assembling simulations rather than programming them.
- Provide a natural learning pathway for increasing simulation complexity
- Provide a redistribution mechanism so that users can both provide and benefit from other pre-built behaviors.

Using the Behavior System enables you to:

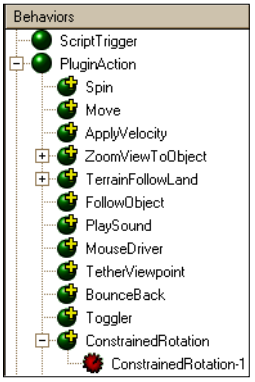
- Reduce the time spent scripting
- Decrease time to prototype
- Build reusable components
- Achieve a modular design for your simulation
- Manage your application's scalability

In essence, the Behavior System provides for visible, efficient, modular, event based design framework for structuring your applications from scratch to ensure scalability as your simulation becomes more complex.

You use the Behavior Workview to interact with the Behavior System in WorldUp, which contains the following core Behavior System components.

The Behaviors Pane

The Behaviors pane is similar to the Type and Nodes panes in the Scene and Type Workviews. The Behavior pane contains a listing of the actual entities (Triggers, Actions, and Grouping nodes) that populate the Behavior System. The Behavior pane displays all of the loaded Behavior types, and groups them based upon their source, including those created by plug-ins and those created by BasicScript (either created by you or imported into your simulation).



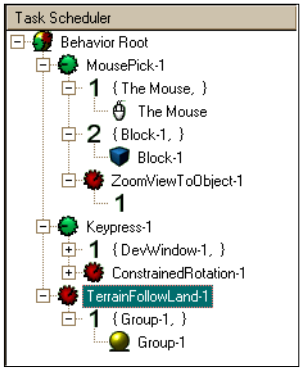
The Behavior Pane

Refer to "The Behavior Object" on page 128 for more information about the structure of these entities.

You use the Behaviors pane to create instances of a particular Behavior type and schedule it in one step by dragging the desired type into the Task Scheduler.

The Task Scheduler

The Task Scheduler shows the execution flow of all scheduled behaviors, as well as each behavior's InputLists and the Inputs they contain. This allows you to both see the execution order of your behaviors as well as drag and drop scene graph objects (Movables) onto specific inputs. In addition, it allows you to change the execution order of your behaviors by dragging them internally within the Task Scheduler.



The Task Scheduler

Refer to "Creating and Scheduling Behaviors" on page 130 for more information.

The Behavior Wizard

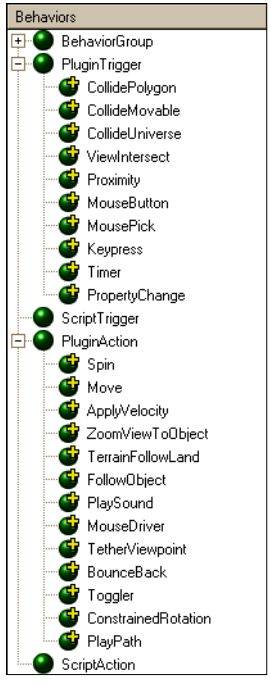
The Behavior Wizard is the interface that enables you to author your own script-based Triggers and Actions. The Behavior Wizard is a series of dialogs that step you through the process of creating a new Behavior type. You access the Behavior Wizard by clicking on the Behavior Wizard button on the Behavior Workview toolbar.

The Behavior WIzard Button

Upon completing the set of dialogs, the wizard generates a new Behavior type as well as a script template you can then modify to define your new Behavior's type's actual behavior. Refer to "Behavior Authoring" on page 132 for more information.

Pre-built Behaviors

WorldUp R5 ships with a basic library of Triggers and Actions most commonly found in interactive simulations. These plug-in behaviors are defined in the TriggerSet1.dll and ActionSet1.dll in your WorldUp R5 plug-ins directory. When WorldUp launches, it detects the presence of these DLLs and registers them with the WorldUp Object System. In the Behavior Workview, these plug-in triggers and actions appear under the PluginTrigger and PluginAction types.



Pre-built Behaviors

A complete description of these pre-built behaviors, how each works, and their parameters can be found in Appendix E, Pre-Built Behavior Library.

For more information about how you can author your own Behavior plug-ins, refer to the *WorldUp Plug-in Author's Guide* that ships with the WorldUp Plug-in Kit (available as a separate module).

The Behavior Object

Strictly speaking, a behavior is some activity that is applied to or demonstrated by any WorldUp Object within the Simulation. This activity is defined by the functions of one or more Trigger and/or Action objects.

Behavior Anatomy

A Behavior is a function that operates on WorldUp objects (nodes, sensors, etc.). It has inputs that act as parameters to the function, and outputs that act as the return value(s) of the function. A Behavior typically reads the input in its Input Slot, makes decisions with that input, writes it's outputs to it's output slots, and fires. These inputs are known as "primary inputs." For each primary input, a behavior has an input slot, an input list, an input description, and a source indicator telling where the behavior is going to get its input from. A behavior can get its input from two sources: either explicitly from its input list or implicitly from the outputs of the behavior above it (its *parent*).

The two significant differences between triggers and actions are:

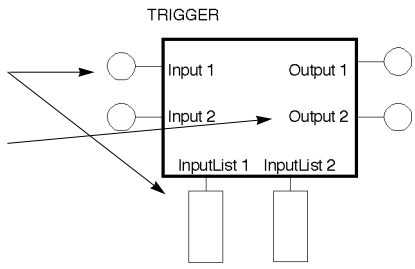
- **Number of Inputs –** A trigger can have 0, 1, or 2 inputs, each of which can be of any WorldUp Object type. An Action has only one input and one output, and it is a Movable.
- **Firing –** A trigger typically has a decision function, which is defined by the user and determines whether or not it fires. An action always fires.

A Trigger's interface consists of primary inputs, primary outputs, and callbacks.

A primary input is a set containing an input slot, an input list, a flag indicating the input's source, and an input description.

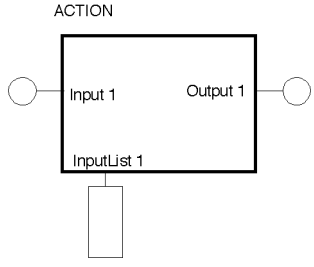
A primary output is an output slot containing the Trigger's current output.

A callback is a function associated with a Trigger that is called in association with a specific WorldUp event, such as Property change or OnCreate.

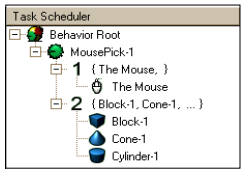


Trigger Anatomy

An Action's interface consists of a single primary input and a single primary output. This is a Movable object. The Movable specified in the input slot is the same Movable the Action will pass to the output slot.



Action Anatomy

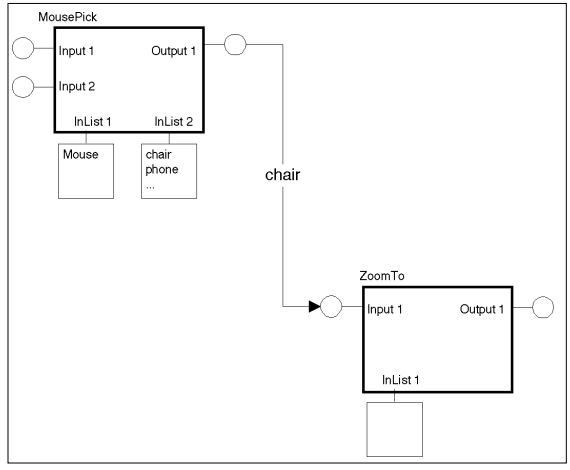


As an example of inputs and outputs, consider a MousePick trigger. This trigger takes two inputs – Input1 is of type Mouse and represents the Mouse object from which you wish to retrieve the screen position. Input2 is of type Movable and represents the list of Movables you consider pickable (for example, in a room, the furniture may be pickable while the walls are not). This trigger has one output – the object that was picked. When the mouse button is pressed, this trigger gets the screen position from the Mouse object in the Input1 slot, and checks to see if it is over any of the objects in Input2.

Assembling Behaviors

It is possible to think of a behavior as an observable activity that emerges from a collection of individual events and responses. For example, a reflex behavior is witnessed as the result of an event (tapping the knee with a hammer) and a response (the calf flexing upwards). WorldUp's Behavior System is designed for this exact purpose of hooking up events to actions to achieve a desired *behavior*. In essence, from a design perspective, a behavior is a dependency network of triggers and actions such that when one trigger fires, it executes all of its children (dependencies). In this manner, a behavior chain can be assembled out of low level triggers (such as, Collide Movable) and actions (such as, Rotate Leg) to achieve a higher-level behavior, such as flexing the calf when a hammer collides with a knee.

Hooking a trigger up to another trigger or action involves *wiring* a trigger's outputs to another trigger/action's inputs. For example, suppose we wish to implement a behavior that says "When the mouse clicks on an object, zoom the viewpoint to the object we clicked on." In this example, our action *zoom viewpoint to a selected object* would need to know what to zoom to. We need to pass the *selected object* from the event *When the Mouse clicks on an object* to the *Zoom To* action so that Zoom To knows which object in fact to zoom to. In this case, we have *wired* the Zoom To action's input to the MousePick trigger's output and made its execution conditional on something being picked.



MousePick Example

In the MousePick example above, we have a MousePick trigger with two inputs. Input 1 is the Mouse from which to retrieve the 2D screen coordinate we will need for our picking. Input 2 is a list of geometries we consider *pickable*. For example, we may have a room in which we want to be able to pick the furniture but not the walls. In this example, the pickables are *chair* and *phone*.

When the user picks chair, our MousePick trigger writes chair to its Output1 slot and fires. Firing causes two things to happen. First, it causes chair to be passed to our ZoomTo behavior. Second, it causes ZoomTo to be executed. ZoomTo has only one input, namely the target to which we will be

zooming. ZoomTo receives chair in its Input1 slot, as well as the fire notification for MousePick, and proceeds to zoom the viewpoint to the chair.

In this example, MousePick inputs are explicitly defined by the user in the trigger's input lists, while ZoomTo uses input implicitly passed from its parent trigger. The source for a behavior's inputs can be one or the other and is defined by the primary input, which is set by you in the Input Definition dialog box.

Creating and Using Behaviors

The Behavior System employs a drag-and-drop environment that allows you to easily assemble conditional action chains out of Trigger and Action primitives to build high-level behaviors that control simulation events and objects. The Behavior Workview is used for all of your behavior management work, including:

- Creating and scheduling behaviors
- Editing behavior inputs and parameters
- Authoring new behaviors
- Importing and exporting script-based behaviors

This section discusses how to create and schedule a behavior and how to edit its inputs and parameters. The following section details behavior authoring with the Behavior Wizard as well as importing and exporting script-based behaviors you or someone else has authored.

Creating and Scheduling Behaviors

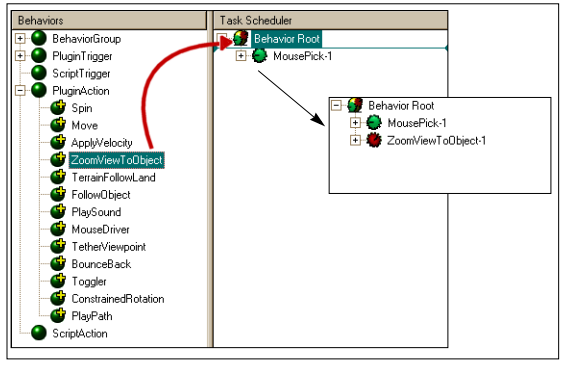
Creating a behavior involves instantiating an object of a particular Behavior type and scheduling it in the Behavior Workview's Task Scheduler. In WorldUp, this is a one step process,

To create and schedule a behavior

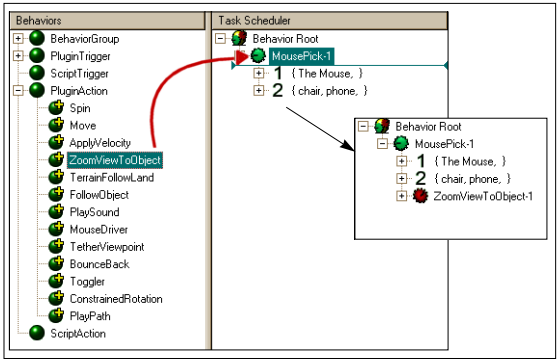
1 Select the type of Behavior you want to create and drag it into the Task Scheduler pane.

2 Drop it onto the object you wish to be its parent.

A *parent* in the Behavior System determines if the behavior you created gets executed every frame (parent is the Behavior Root) or if its execution is conditional upon another behavior, as discussed in the On Mouse Pick Zoom To behavior example in the preceding section.



Dragging a behavior onto the Behavior Root object inserts the behavior as a child of the Behavior Root, placing it at the root level of the Task Scheduler. All root-level behaviors are executed every frame.



Dragging a behavior onto another behavior inserts the new behavior as a child of the behavior onto which it was dropped. This schedules the new behavior to be executed when its parent behavior fires. In the example, when MousePick fires, ZoomViewToObject executes.

The order of execution can be critical in an application. For example, you should typically move your object before checking for collisions. Doing so afterward may find your object halfway inside another.

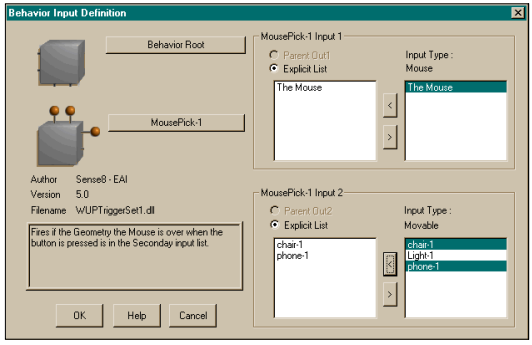
To change a behavior's order of execution

➤ Drag it internally in the Task Scheduler.

Note The same dragging conventions apply here as they do in the Scene Graph pane. Once you have dropped your behavior onto it's new parent, you will again be prompted to edit your behavior's inputs.

Editing Behavior Inputs

When you drop a behavior into the Task Scheduler, you are prompted to supply the inputs for the specified Behavior. This is indicated by the Behavior Input Definition dialog box.



The input boxes on the right sight of the dialog allow you to indicate to WorldUp whether your behavior will be acting upon a list of objects explicitly specified by you (Explicit List) or acting upon an output supplied by it's parent behavior (Parent Out).

Typically, behavior convention suggests that the first input to a behavior (input 1) is the primary input the behavior acts upon. For simple actions such as spin or move, the object to act upon can be very clear. The second input is typically referred to as the object(s) the behavior acts with. Many actions, have no second input. Most triggers, however, do have second inputs. For example, MousePick takes as Input 1 the Mouse object it is going to get its screen position from, and Input2 as the list of objects it will check for intersection with.

The convention for behavior authors is to apply the Cartesian product to the two sets of inputs, such that if MousePick had multiple mice in Input 1, for example Mouse-1 and Mouse-2, the execution would be:

If Mouse-1 picked chair-1 Then Fire

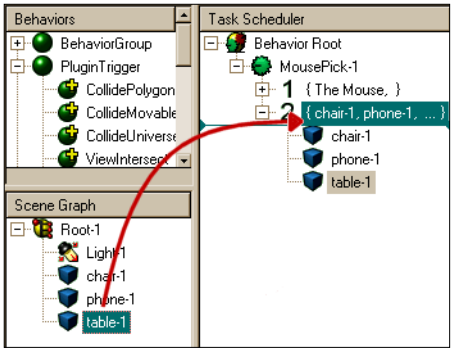
If Mouse-1 picked chair-2 Then Fire

If Mouse-2 picked chair-1 Then Fire

If Mouse-2 picked chair-2 Then Fire

Note This is only a convention. You are free to use the inputs in whichever manner you choose. The only rule WorldUp applies is that it will execute a given behavior for each object in that behavior's Input1. This also means that a behavior with no inputs will not be executed.

In addition to using the Behavior Input Definition dialog box, a simpler way of adding inputs to a Behavior is by simply dragging them from the Scene Graph pane onto the input number of the Behavior you wish to add it to. It is not enough to simply drag it onto a behavior, since it is not clear which InputList you wish to add it to.



Behavior Authoring

Authoring behaviors is a significantly different process than using behaviors. Your goal as a behavior author is to create a perfectly encapsulated set of functionality that exposes itself through a set of parameters that can then be modified interactively. WorldUp provides two mechanisms for authoring behaviors: the Behavior Wizard and the Plug-in Kit. The tool you choose will depend on your programming background and preferred programming environment (C versus BasicScript) and the requirements of the behavior (licensing, performance, eedistributability, OpenGL, etc.).

This section covers the BasicScript tools for authoring behaviors, as well as importing and exporting behaviors. For more information on using the WorldUp Plug-in Kit to author behaviors, refer to the *WorldUp Plug-in Author's Guide*.

Using the Behavior Wizard

A behavior script consists of a Behavior type and a BasicScript callback function. All script-based behaviors are authored through the Behavior Wizard interface. This interface automatically creates the type and the script for you, as well as provides you with a code skeleton that you can use to begin authoring your behavior. If you are subtyping an existing behavior script, the Behavior Wizard will insert that the base type's code as the skeleton base (if the code is not protected).

To create a new behavior script using the Behavior Wizard

1 Select the Behavior type you wish to subtype in the Behaviors Workview.

If you select an existing script-based action or trigger, the Behavior Wizard assumes you wish to subtype the selected behavior. Otherwise, it creates a new behavior from scratch.

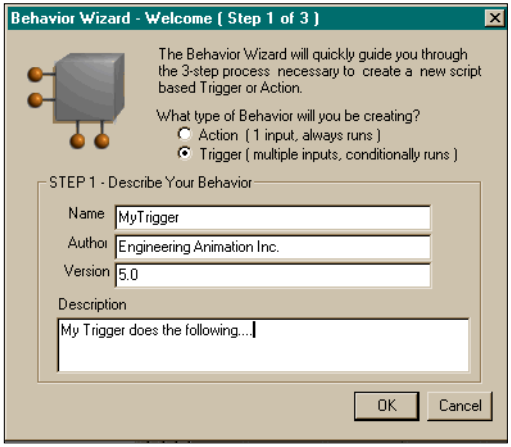
2 Click the Behavior Wizard  button.

The Behavior Wizard is a series of dialogs aimed at simplifying the creation of a new Behavior type. Upon completing the set of dialogs, the Behavior Wizard generates a new Behavior type as well as a script template that you can then

modify to either define your new Behavior's type's actual behavior, or customize an existing behavior.

3 Name and describe your new behavior.

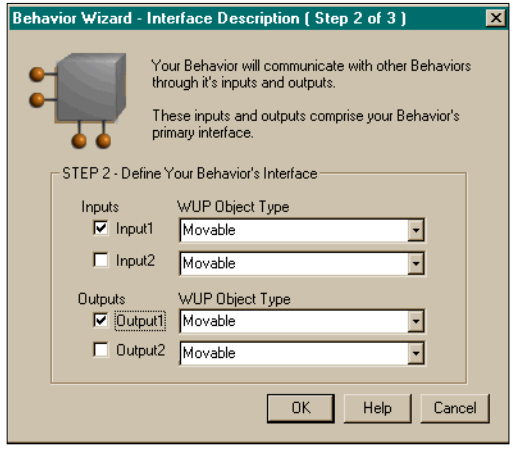
The first step in the Behavior Wizard is to indicate to WorldUp whether you are creating an Action or a Trigger. If you are subtyping an existing script-based behavior, this will already be checked.



Name and Describe your Behavior

4 Define your behaviors Inputs and Outputs.

The second step asks you to define the number of inputs and their types.



Define the number and types of your inputs and outputs

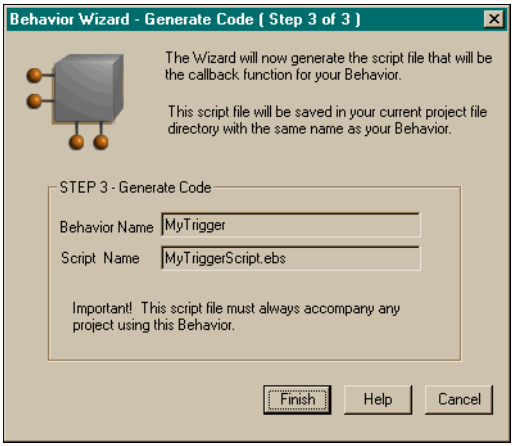
If you are creating an action, this will automatically be set to 1 input of type Movable and 1 output of type Movable. This is according to the definition of an action as covered in "Behavior Anatomy" on page 128.

If you are creating a new trigger, you have a choice of the number of inputs and outputs and their types.

If you are subtyping an existing script-based trigger, these will be filled in based on the parent types inputs and outputs. Since the ensuing script depends on these, it is advised you not change them unless you plan on changing the script as well.

5 Finish, confirm, and create.

Finally, WorldUp attempts to create the script type and a new script file. Assuming you have provided a unique name for your new type, and WorldUp has proper disk access, a final confirmation dialog appears.



Finish, confirm, and create your script

Clicking Finish adds this type to your list of script-based triggers (or actions, depending on what you chose to subtype). It also opens the new script file ready for editing.

This script and type are now part of your project and will automatically be loaded the next time you open your project. Therefore, it is crucial you keep your new script in one of the locations specified by your Scripts directory path settings.

Customizing Your New Script

The behavior script generated by the Behavior Wizard is referred to as a *Script Handler* or callback function. This is the routine that gets called every frame for each object in the behavior's InputList1. behavior Script Handlers typically have the following structure:

```
Subroutine Begin (Trigger)
  Read Trigger's Inputs
  Process Data
  Write Trigger's Outputs
  Fire Trigger (or stop trigger)
Subroutine End
```

For example, here is a Behavior Wizard generated trigger Script Handler:

```
sub task(mytrigger as CollideMovable)

  dim flag as boolean

  'READ INPUTS
  dim ActUpon as Movable
  set ActUpon = mytrigger.in1
  dim ActWithList as List
  set ActWithList =
  mytrigger.InputList2
  dim ActWith as Movable
  set ActWith = CastToMovable
  (ActWithList.GetFirstObject())


  'TODO: Insert decision processing
  code here.
  While ActWith is not nothing
    success =
    ActUpon.IntersectsMovable(ActWith
    )
    if success then
      'WRITE OUTPUTS
      set mytrigger.out1 = ActUpon
      set mytrigger.out2 = ActWith

      'FIRE TRIGGER
      flag = mytrigger.Fire()
    End If
    Set ActWith =
    ActWithList.GetNextObject()
  Wend
end sub
```

A behavior's Script Handler can now be treated like any other script, with the exception that the parameter type passed into the script must be the same as the behavior's actual name.

To edit the script

1 Select the behavior in the Behavior Workview.

2 Click the Edit Script button.

Some additional conventions to remember when working with script-based behaviors:

- If the trigger is a two-input trigger, InputList2 is assumed to be handled by the trigger author.

  The benefit of this approach is that the Task Scheduler does not automatically define the relationship between Inpulist1 and Inputlist2 (for each; for each). You can define it how you please. For example, if you wanted to only fire for the *first* object collided with, you could easily define this relationship by rearranging the loop.

```
set ActWith =
CollidableList.GetFirstObject()
success = False
While not Success
  success =
  ActUpon.IntersectsMovable(TargetMov
  able)
  if success then mytrigger.Fire()
  set TargetMovable =
  CollidableList.GetNextObject()
Wend
```

- By convention, the script file created bears the same name as the behavior type created. The system also associates the script with the behavior type, acting essentially as a callback for that behavior type. As such, it is very important that a script-based behavior's callback script be locatable by the system during project loading. You should never rename a behavior's script file.

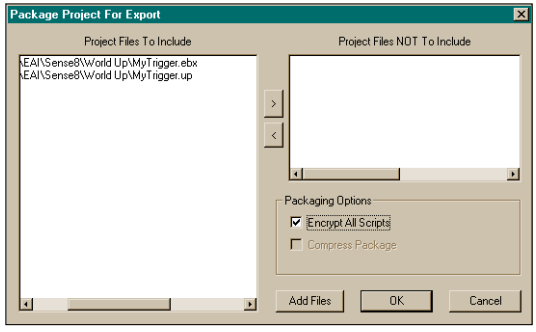
Importing and Exporting Script-Based Behaviors

Behavior's authored with the WorldUp Plug-in Kit are automatically detected and loaded when WorldUp starts or a simulation is loaded. With scripted behaviors, you must import and export them explicitly.

To export a script-based behavior

1 Select the Behavior type you wish to export in the Behaviors Workview.

2 Click the Export Behavior button.



Exporting an Encrypted Behavior Script

This exports your behavior into a behavior file (.pup). This file is a collection of your selected behavior types and their callback scripts, along with any additional files you wish to add. These files are combined together into a single PUP file during export, at which time you can also decide whether or not to encrypt the scripts for redistribution without giving away the source code.

To import a script-based behavior into your project

1 Select the Import Behavior button on the Behavior Workview.

2 Select available any PUP file.

This brings the types and scripts into your project in a subdirectory of the same name as the PUP file. This behavior is now part of your project and will be saved with it. It is no longer necessary to keep the PUP file.

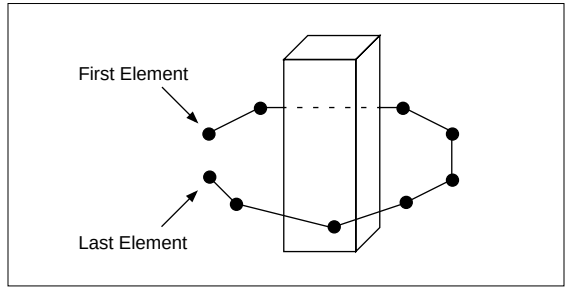
Note If the subdirectory contains additional files, such as sounds, textures, or models, you need to add these to your search path.

15

Paths

Paths are a series of position and orientation records that you can use to guide Viewpoint or Movable objects. In the case of Movable objects, the position and orientation information in the path affects the object in its parent's reference frame. You can dynamically create, record, edit, save, load, and play back paths in a variety of ways. You can also use interpolation to smooth a roughly defined path.

As shown in the figure below, paths are made up of a set of discrete elements, where each element stores a specific position and orientation.



A path Around an Object

These elements are stored in .PTH files which are referenced by WorldUp Path objects. You record elements by specifying the Viewpoint or Movable object whose position and orientation you want to record and then manipulating that object’s position and orientation as you record. You can record continuously or one element at a time, adjusting the

Viewpoint or Movable object's position as you go. A path's playback is not limited to the object from which it was recorded. Any Viewpoint or Movable object can play any recorded path. So, for example, you could record a path from a viewpoint, then have a graphical object follow that path.

Paths are useful for a variety of applications. For example, if you are creating a demonstration program, you can record an optimal path through the virtual environment before the actual demonstration. Viewpoint paths are useful for any application in which it may be important for the user to see certain aspects of the virtual world. You can also use viewpoint paths whenever an application requires that a viewpoint be moved from one location to another and you wish to provide a smooth transition.

Similarly, you may have many uses for having Movable objects follow paths. Consider a simple case in which you want to have a door swing open and shut. You could attach a sensor, such as a Spaceball, to the door, and while twisting the Spaceball to open the door, record the door's path. Through scripts, you could then indicate when the path is to be played, for example, each time the user clicks the Door object. By setting the path's playback mode to Oscillate, the path will play to the end, then reverse direction, causing the door to close.

Many of the properties of a Path object can be controlled through the Path Browser. For information on additional Path properties, such as Speed, search on Path Type in the online Help.

Creating New Paths

You can either create a new, empty path which contains no elements, or you can create a new, interpolated path which contains elements that are interpolated from elements of an existing path.

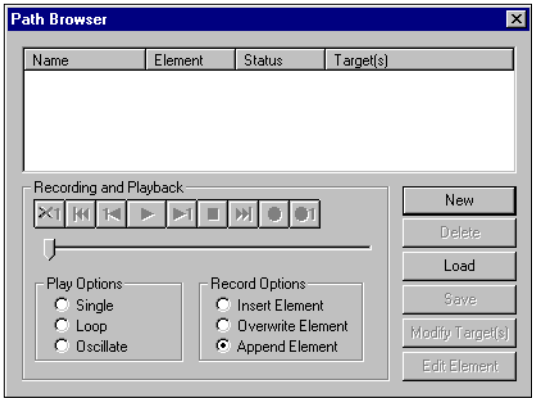
Creating a New Empty Path

This section explains how to create a new path containing no elements.

To create a new, empty path

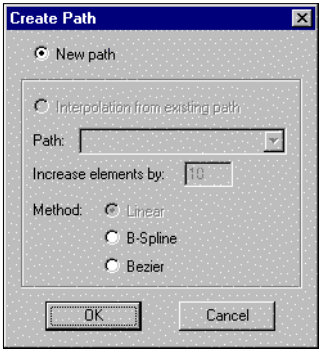
1 Click the Browser - Toggle Path  button.

The Path Browser displays.



2 In the Path Browser, click New.

The Create Path dialog box displays.



3 Accept the default option, New Path and click OK.

A new Path object appears in the Type Workview and a new path is added to the Path Browser.

Note You can also create Path objects for empty paths from the Type Workview as described in "Creating an Object" on page 87.

Creating a New Interpolated Path

This section explains how to create a new path containing elements interpolated from an existing path. Your project must already contain one or more recorded paths before this procedure becomes available.

To create a new, interpolated path

1 Click the Browser - Toggle Path button.

The Path Browser displays.

2 In the Path Browser, click New.

The Create Path dialog box displays.

3 Click Interpolation from Existing Path.

4 In the Path text box, select the existing path (which contains two or more elements) from which you want to interpolate the new path.

The selected path will be unaffected by the interpolation.

5 In the Increase Elements By box, type the number of interpolated elements that you want to insert between each element of the original path.

6 Click the desired interpolation method, as described below:

- *Linear* – a straight line path between elements.
- *B-Spline* – a curve which is the *smoothest* of all the options, but which does not in general pass through the elements of the original path.
- *Bezier* – a smooth curve which passes through the elements of the original path.

Note The orientations of the elements are also interpolated, however the method used to interpolate orientations is always linear, independent of the method chosen to interpolate positions.

7 Click OK.

A new Path object appears in the Type Workview and a new path is added to the Path Browser.

Using Existing Paths

In the Path Browser, you can load an existing .PTH file and a Path object is created that references that file.

To load an existing path

1 Click the Browser - Toggle Path button.

The Path Browser displays.

2 In the Path Browser, click Load.

3 In the Open dialog box, locate and open the PTH file that you want to load.

A Path object is created in the Type Workview and the loaded path is added to the Path Browser.

Note When you load a path, you are only loading the information contained in the PTH file. Information that is stored in the Path object for which the PTH file was originally created, such as the targets for the path (see below), is not loaded.

You can also load paths by dragging a PTH file from any file browser directly into the Development window.

Moving Viewpoints and 3D Objects Along Paths

Before you can record or play a path, you must specify the Viewpoint or Movable object from which you want to record the path (known as the Record From target), and the Viewpoint and/or Movable objects to which you want to play the recorded path (known as the Playback targets). These associations are known as *motion links*.

The instructions below describe how to link paths to target objects within the Path Browser. You can also create motion links for paths from the Motion Link Sources and Motion Link Targets dialog boxes, as described in "Motion Links" on page 144.

To specify Playback and Record from targets

1 In the Path Browser, double-click the path whose targets you want to set, or single click the path and click the Modify Target(s) button.

The Path Targets dialog box displays.

2 In the Potential Targets box, select a Viewpoint or Movable object that will follow the path when the path is played.

3 Click Add to add the selected object to the list of Playback Targets.

You can play a path on multiple objects. So, if necessary, repeat this procedure to add all of the desired objects to the Playback Targets list.

4 To remove an object from the Playback Targets list, select the object in the Playback Targets list and click Remove.

5 In the Record From Target box, select the Viewpoint or Movable object whose position and orientation you want to record to create the path's elements.

The only objects available are those that you have specified as Playback targets. Thus, for example, if you want to record from Viewpoint-1, but you only want to play the path on Block-1, you will have to add Viewpoint-1 as a Playback target in order to be able to record from it, and then later remove Viewpoint-1 when you are ready to play the path.

6 Click Done to close the Path Targets dialog box.

For each object that you specify as a target of a path, WorldUp automatically creates a corresponding MotionLink object. When you remove a target, the corresponding MotionLink object is automatically deleted.

For more information on MotionLink objects, see "Motion Links" on page 144.

Recording Paths

Once you have specified a Record From target for your path, you can record elements. You can record elements continuously or one element at a time.

To record paths continuously

1 Ensure that you have specified a Record From target as described on page 140.

2 In the Path Browser, select the path that you want to record.

Note You can record multiple paths at once. To select multiple paths, press the CTRL key while you click each path.

3 Click the Record button.

4 In the Development window, manipulate the Viewpoint or Movable object that you specified as the Record From target for the selected path.

5 Click the Stop button to stop recording.

For information on how to edit recorded elements, see "Editing Path Elements" on page 142.

To record a single element for the path

1 Ensure that you have specified a Record From target as described on page 140.

2 In the Path Browser, select the path for which you want to record an element.

Note You can record multiple paths at once. To select multiple paths, press the CTRL key while you click each path.

3 In the Record Options section of the Path Browser, select one of the options as described below:

- *Insert Element* – The recorded element will be inserted just before the current element.
- *Overwrite Element* – The recorded element will replace the current element.
- *Append Element* – The recorded element will be appended at the end of the path.

4 If you have chosen the Insert Element or Overwrite Element record option, drag the slider below the Recording and Playback buttons so that the path is positioned at the location where you want to record the element.

As you drag the slider, the Element column of the Path Browser indicates the current position of the path.

5 In the Development window, set the desired position and orientation of the object that you have designated as the Record From target.

6 Click the Record One Frame button.

7 Continue manipulating the Record From object and recording elements to achieve the desired path.

Note Keep in mind that when creating paths by recording single elements, it is often useful to record rough paths, composed of just a few elements, and then interpolate a smoother path from the one you recorded. See "Creating a New Interpolated Path" on page 139 for instructions.

For information on how to edit recorded elements, see "Editing Path Elements" on page 142.

Playing Paths

You can play paths from the Path Browser as you develop your application, or you can play paths from scripts.

Each Path object has a Forward property in the Property pane. The default value is True, meaning that the path will begin playing in the forward direction. The discussion below assumes that this property is set to True.

To play paths from the Path Browser

1 Ensure that you have specified at least one Playback target, as described on page 140, for each path that you want to play.

2 In the Path Browser, select the path that you want to play.

Note You can play multiple paths at once. To select multiple paths, press the CTRL key while you click each path.

3 In the Play Options section of the Path Browser, select one of the options as described below:

- *Single* – The path will play to the end and then stop.
- *Loop* – The path will play to the end and then repeat at the beginning.
- *Oscillate* – The path will play to the end and then reverse direction when the final element is played.

4 If the path is currently positioned at the final element, click the Rewind button.

5 Click the Play button.

6 Let the path play to the end (if you chose the Single playing option), or click the Stop button to interrupt the play.

7 To play one frame (one element) of the path, click the Forward1 or Back1 button, depending on which direction you want the path to play.

To play paths from scripts

➤ Call the Play or Play1 WorldUp function.

Keep in mind that each Play Option in the Path Browser has a corresponding property in the Property pane for all Path objects. You can set these properties from scripts.

For information on writing scripts, see the *WorldUp Programmer's Guide*. For information on the Play and Play1 functions, search on Play or Play1 in the online Help.

Editing Path Elements

From the Path Browser, you can modify or delete any element in your path.

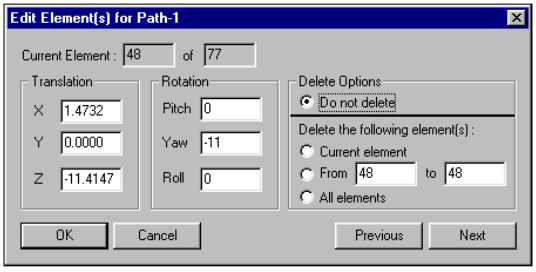
To modify the translation and rotation of path elements

1 In the Path Browser, select the path containing the elements you want to modify.

2 Use the slider to position the path at the desired element.

3 Click the Edit Element button.

The Edit Element(s) for Path-1 dialog box displays the translation and rotation values of the current element.



4 To focus on another element, click the Previous or Next buttons to move through the path.

The number of the element that currently has focus is listed at the top of the dialog box.

5 To modify the element's translation, type new values in the X, Y, and Z boxes under Translation.

6 To modify the element's rotation, type new values in the Pitch, Yaw, and Roll boxes under Rotation.

7 Click OK to apply the changes.

To delete elements from a path

1 In the Path Browser, select the path containing the elements you want to delete.

2 Use the slider to position the path at the desired element.

Note This is not necessary if you intend to delete all elements.

3 Click the Edit Element button.

The Edit Element(s) dialog box displays.

4 To focus on another element, click the Previous or Next buttons to move through the path.

The number of the element that currently has focus is listed at the top of the dialog box.

5 In the Delete Options section, click one of the following options:

- *Current Element* – Deletes the current element only.
- *From ... To ...* – Deletes the specified range of elements. Type the numbers of the first and last elements of the range you want to delete.
- *All Elements* – Deletes all elements in the path.

6 Click OK to delete the specified element(s).

Saving Paths

If you do not save your recorded paths, then no PTH files are created for them. Thus, while the Path object remains, all recorded elements are lost once the project is closed.

Note It is not necessary to save an interpolated path if the path from which it was interpolated has been saved. Each time you reopen the project, the interpolated path is recreated.

To save paths

1 In the Path Browser, select the path that you want to save.

2 Click Save.

3 In the Save As dialog box, select the directory in which the current UP file is located and specify a name for the PTH file.

Note If your PTH files are located in any directory other than the UP file's directory, the project will not be able to locate the PTH files when the project is opened again.

Deleting Paths

To delete a path

1 In the Path Browser, click the path that you want to delete.

Note You can delete multiple paths at once. To select multiple paths, press the CTRL key while you click each path.

2 Click Delete.

The Path object is deleted, along with any MotionLink objects for which the deleted path was the source.

Note You can also delete paths from the Type Workview, as described in "Deleting an Object" on page 88.

Paths and Sensors Use Motion Links

Understanding Motion Links can be very helpful when using paths. When you specify Record From and Playback Targets in the Modify Targets dialog box, WorldUp creates the underlying MotionLink objects used in your simulation in the Type Workview.

You can control any valid Motion Link Target with any valid Motion Link Source. Valid Motion Link Sources are Paths, Sensors, and Movables.

Note It is actually feasible, though not recommended, to specify any WorldUp object as the source, since the property is of type VBase. However, objects would need to have Rotation and Translation properties to have any effect on the Target.

Valid Motion Link Targets are Movables and Viewpoints.

Sensors

Sensors are devices, such as a mouse or a Spaceball, that can be used to manipulate Movable objects (such as geometries) or change your viewpoint in the universe.

In the Type Workview, the Sensor object type contains subtypes for each sensor that WorldUp supports.

To use a sensor, you must attach the hardware to your computer, create a Sensor object under the appropriate Sensor type, and create motion links between the sensor and the Viewpoint and/or Movable objects in the universe.

By default, WorldUp provides each new universe with a Mouse object (The Mouse). It also provides a default Viewpoint (Viewpoint-1) and MotionLink (MotionLink-1), which links The Mouse to Viewpoint-1. This allows you to instantly begin navigating the viewpoint of the default Development window (DevWindow-1) with the mouse. If you are using a type of sensor other than a mouse, you'll need to create your own Sensor object and MotionLink object.

When you delete a Sensor object, all MotionLink objects for which the deleted Sensor object was the source are also deleted.

For information on how to configure specific sensors, see Chapter 17, *Using Input Devices*. For information on how to use motion links to link your sensor to Viewpoint and Movable objects, see the Motion Links section below.

Motion Links

Sensors and paths allow you to interact with a virtual world by providing you with control over the motion of Viewpoint and Movable objects. To associate a

sensor or path with an entity in a world, use motion links. A motion link connects a *source* of position and orientation information (a Sensor or Path object) with a *target* (a Viewpoint or Movable object) that moves to correspond with that changing set of information.

Through properties of MotionLink objects, you can specify such attributes as whether the motion link will be enabled during development, while the simulation is running, neither, or both. Also, you can specify in which reference frame the source object is to manipulate the target object. For a description of properties specific to MotionLink objects, search on MotionLink Type in the online Help.

Assigning Motion Link Targets

To assign a target to a source object

1 In the Type Workview, select the Sensor or Path object that you want to be the source of a new motion link.

2 Select Edit Motion Link Targets from the Objects menu, or right-mouse click the object in the Type Workview and select Edit Motion Link Targets.

The Motion Link Targets dialog box appears.

3 In the Potential Link Targets box, click the Viewpoint or Movable object that you want to be the target of the motion link and click Add.

4 Repeat this procedure, if necessary, to link additional target objects to the source.

A MotionLink object is created in the Type Workview for each target that you add.

5 Click Done when you are finished.

Note If the source of the motion links you are creating is a path, be aware that the targets you specify from this dialog box will be Playback targets only. You will have to specify the Record From target from the Path Browser as described on page 140.

Assigning Motion Link Sources

To assign a source to a target object

1 In the Type Workview, select the Viewpoint or Movable object that you want to be the target of a new motion link.

2 Select Edit Motion Link Sources from the Objects menu, or right-mouse click the object in the Type Workview and select Edit Motion Link Sources.

The Motion Link Sources dialog box appears.

3 In the Potential Link Sources box, click the Sensor or Path object that you want to be the source of the motion link and click Add.

4 Repeat this procedure, if necessary, to link additional source objects to the target.

A MotionLink object is created in the Type Workview for each source that you add.

5 Click Done when you are finished.

Note If you select a Path object as a source, be aware that the target object will be a Playback target only. You will have to specify the Record From target from the Path Browser, as described on page 140.

Removing Motion Links

You can remove motion links by deleting the appropriate MotionLink object from the Type Workview, or by removing targets or sources.

To remove sources or targets

1 In the Type Workview, click the object that is either the source or target of the motion link you want to delete.

2 Select Edit Motion Link Sources or Edit Motion Link Targets from the Objects menu, depending on which type of object you have selected.

You can also right-mouse click the object in the Type Workview and click the corresponding option.

The Motion Link Sources or Motion Link Targets dialog box displays.

3 In the Linked Sources or Linked Targets box, select the object you want to remove and click Remove.

4 Repeat this procedure, if necessary, to delete additional objects.

For each object that you remove, the corresponding MotionLink object is deleted from the Type Workview.

5 Click Done when you are finished.

To delete MotionLink objects directly

1 In the Type Workview, expand the MotionLink object type to see its existing objects.

2 To determine which MotionLink object you want to delete, check the current values of the Source and Target properties for the object.

3 Click the Delete Selected  button on the Type Workview.

Motion Link Properties

A MotionLink takes a source of position and orientation information (a sensor or path object) and applies it to a target (a viewpoint or movable object).

WorldUp creates a default viewpoint (Viewpoint-1) and a default Mouse object (The Mouse), and connects these two objects with a default Motion Link (MotionLink-1). If the motion link is enabled (its Enabled property is set to True) then WorldUp automatically updates the target of the motion link with the sources input.

MotionLink Property	Definition	Acceptable Value(s)
Application Active	Indicates whether the link will be active when the application runs.	True – The MotionLink is applied at runtime False – The MotionLink is not applied at runtime
Development Active	Indicates whether the link will be active when you are developing the application.	True – The MotionLink is applied while developing False – The MotionLink is not applied while developing
Enabled	If set True, then the target of the MotionLink is updated with source input.	True – Update source False – DO NOT update source
Name	The name of the MotionLink in the simulation.	Any string value

MotionLink Property	Definition	Acceptable Value(s)
Reference Frame	This is the reference frame in which the target of the MotionLink will operate.	Local – Target operates in its local reference frame. World – Target operates in the world reference frame Parent – Target operates in its parent's reference frame Viewpoint – Target operates in the viewpoint reference frame
Source	The source object of the MotionLink can be either a Sensor object such as the mouse or a Path object.	Any valid device or path object
Target	The target of the MotionLink can be a Viewpoint or Movable object.	Any valid viewpoint or movable object
Tasks	Name(s) of the script(s) to be executed once per frame during the simulation.	Any valid script. The script must contain a task subroutine. The task will be run once each frame of the simulation.

16

Sounds

WorldUp can play sounds from WAV files. You can use scripts to control how and when a sound plays.

To play sounds, the machines on which you plan to develop or deploy your simulation must have the following:

- A sound card
- An operating system configured to use the sound card
- A pair of speakers attached to the sound output ports

Be sure to set your sound path (using the File Access Settings on the Options menu) and put the sounds you use in your simulation in the sound folder. See "Configuring Directory Paths" on page 39.

Creating a Sound Object

You can create a Sound object using drag-and-drop or from the Type Workview.

To create a Sound object using drag-and-drop

- Drag a wave file (.WAV) from your file browser onto WorldUp.

A new Sound object is created and is named to match the filename of the sound that you dragged.

To create a Sound object in the Type Workview

1 In the Type Workview, select the Sound object type.

2 Click the Instantiate Selected Type  button.

The File Open dialog box appears.

3 Navigate to the sound file you want WorldUp to play and click Open.

Once the sound object is created, you can test your sound.

To play your sound

1 Select the Sound object.

2 Set its Playing property to True.

3 Click the Run in DevWindow  button or the Run in AppWindow  button.

Changing Sounds

To specify another sound file

1 In the Type Workview, select the Sound object you wish to replace.

2 In the Property pane, select the Editable tab.

3 Set its Playing property to True, if not set already.

4 Double-click the Filename property.

The File Open dialog box appears.

5 Navigate to the sound file you want WorldUp to play and click Open.

6 Click theRun in DevWindow  button or the Run in AppWindow  button to hear your new sound.

Finding Sounds for your Application

A number of good sound editors are available for free over the Internet. Most of the sample sounds for WorldUp were created with a microphone, office supplies, and a simple sound editor for cutting and pasting sound samples.

You can also find sound libraries on CDs at any large computer software store.

Changing Sound Properties

You may want to modify the Sound object's properties and/or the Audio properties of the Universe object to control factors such as which sound device to use or whether the sound plays continuously.

Search on Sound Type in the online Help for a description of the unique properties for the Sound object type. Search on Universe Type for a description of the various Audio properties of the Universe object.

Using Scripts to Play a Sound

If a Sound object's Playing property is set to True, the sound automatically plays when you run the simulation. To allow the sound to play continuously, the Repeat property must also be set to True. If you want to play the sound under certain conditions only, write a script using WorldUp's Play command and attach that script to your Sound object.

See the *WorldUp Programmer's Guide* for information on how to write and attach a script to an object.

Setting the Audio Listener Viewpoint

The Universe object contains a pre-defined property called Audio: Listener. This property indicates which viewpoint represents the listener. In other words, it represents where the **ears** of the viewer are positioned in the universe.

By default, the Audio: Listener property is set to Viewpoint-1.

If you set the application window to use another viewpoint, you also need to set the Audio: Listener property to that viewpoint, so sounds can be spatialized correctly.

Troubleshooting Sounds

This secton provides some troubleshooting steps to take if you are having trouble with your sounds.

Sounds won't play

The following is a list of reasons why sounds might not play:

- Does your sound system work?

  Make sure that your sound system is properly set up. Use the sound player that comes with your operating system to try to play a sound. If the sound doesn't play, it could be that your sound drivers are not installed properly, your sound card is not working, or your system volume is simply turned all the way down. Check your system's sound settings.

- Does the sound have its Playing property set to True?

  To play a sound, the Sound object's Playing property must be set to True. You can set this directly in Property pane of the Type Workview or by calling the script method Play. (The command would be `snd.Play`, where `snd` is a variable pointing to a sound.)

- Is something hogging the sound device?

  Another application might be running that's holding on to the sound device, not letting WorldUp play its sounds. Close the other application and try again.

- Is Audio: Listener set correctly?

  The Audio: Listener property of the Universe object should be pointing to the correct viewpoint (whichever viewpoint is used in the main application window).

- What isAudio: Rolloff set to?

  The Audio: Rolloff property of the Universe object determines how fast the sounds in space fall off. If the number is too low for the size of your universe, sounds might be falling off too quickly.

Sounds are not spatialized correctly

- Is the Attached To property set to the correct object?

  Make sure the Sound object's Attached To property is set to a valid object which is still in your scene.

- Is Audio: Listener set correctly?

  The Audio: Listener property of the Universe object should be pointing to the correct viewpoint (whichever viewpoint is used in the main application window).

- What isAudio: Rolloff set to?

  The Audio: Rolloff property of the Universe object determines how fast the sounds in space fall off. If the number is too high for the size of your universe, you might not, in the area in which you move in your universe, be able to tell the difference in roll-off. Try decreasing this number and see if the results are better.

My sound is playing too fast or too slow

If you are using the DiamondWare sound system (this is the default for WorldUp on the Windows platform – check the Audio Device property on the Universe object), sounds must be recorded at a frequency of 22 K. Any sound editor will allow you to resample a sound file.

Make sure the sound's Pitch property is set to 1.0.

17

Using Input Devices

Sensor objects in WorldUp generate position, orientation, and other kinds of data by reading input information that originates in the real world. You can use the input to control motion and other behavioral aspects of objects in the simulation. Sensors permit the user of a WorldUp simulation to be directly coupled to the Viewpoint objects and Movable objects in the universe. By linking a specific Sensor object to a viewpoint, you can use that sensor to navigate. By linking a specific Sensor object to a Movable object, you can use that sensor to control the specified object.

WorldUp provides Sensor properties that allow you to access and, in some cases, modify the state of a sensor. For example, you can retrieve the current translational and rotational value for a sensor, control the scale factors of the translation and rotation records, and retrieve miscellaneous data, such as button press events.

Many of the 3D and 6D (position/orientation) sensors that are available are supported by WorldUp. There are two principal classes of such sensors: desk-based sensors and sensors that are worn on the body. While most desk-based sensors generate relative inputs, that is, *changes* in position and orientation, devices worn on the body typically generate absolute records, that is, values that correspond to their specific spatial location.

Desk-based sensors are conventional devices, such as a mouse, a joystick, or an isometric ball. The Logitech Space Control Mouse (Magellan) and Spacetec IMC's Spaceball are isometric balls that respond to forces and torques applied by the user. Using such devices, users can directly manipulate, displace or rotate a 3D object – the object acts like it is directly connected to the sensor. Isometric ball sensors are also useful for moving the viewpoint; the applied displacements and rotational forces move and rotate the viewpoint. In this mode of operation, with an isometric ball sensor attached to the viewpoint, the sensor operates like a fly-by-wire helicopter.

Sensors worn on the body (sensors that generate absolute records) include *electromagnetic* 6D trackers such as the Polhemus FASTRAK and Ascension Bird. You can use this type of sensor for viewpoint tracking when it is attached to a head-mounted display. Some devices, like the Virtual i-O i-glasses! and the StereoGraphics CrystalEyes VR, provide left and right video displays combined with head-tracking capability. In addition to electromagnetic devices, a variety of *ultrasonic ranging/triangulation* devices and *optical* devices exist for absolute position and orientation tracking. One example is the ultrasonic Logitech 3D Mouse and Head Tracker.

Regardless of their underlying hardware technology, WorldUp Sensor objects are treated similarly and can be used interchangeably in an application. Once a sensor object is created, it is automatically maintained by the simulation manager, so you do not have to deal directly with considerations such as whether the sensor is returning relative or absolute records, or whether it is polled or streaming its data.

WorldUp provides drivers for the devices listed below, making them easy to connect to your computer and use in your applications.

For the most up-to-date information about sensors supported by WorldUp, contact SENSE8 Technical Support, or check our Sensor Setup web page at:

http://www.sense8.com/sensorsetup/index.html

This web page shows what devices are supported and how to set up the devices correctly. See "Technical Support" on page 10 for contact information.

In addition to support for the devices shown above, WorldUp provides functions for easily obtaining input from the keyboard (search on GetKey() in the online Help).

Creating Sensor Objects

To use sensors in your WorldUp simulation, you must attach the sensor hardware to your computer, configure the sensor, create a Sensor object in WorldUp from the appropriate Sensor object type, and then link that Sensor object to the appropriate Viewpoint object or Movable object.

To create a sensor object

1 Attach the sensor hardware to your computer.

2 Configure the sensor to run on your computer (set the DIP switches, display settings, etc.).

"Working With a Specific Sensor" on page 156 describes each sensor and any special configuration instructions.

3 In the Nodes pane, click the Sensor subtype from which you want to create your Sensor object. There is a separate type for each supported sensor.

"Working With a Specific Sensor" on page 156 describes each supported sensor and indicates which WorldUp object type to use for that sensor.

4 In the Property pane, set the available properties as appropriate.

For a complete reference of all WorldUp properties, search on “Property Reference” in the online Help.

- *Serial Baud Rate* – This property is applicable to serial sensors only. The baud rate will be set to the default value that WorldUp uses. The configuration procedures described in "Working With a Specific Sensor" on page 156 assume the default baud rate. So, if you want to use a different baud rate you need to change the configuration accordingly.
- *Serial Port* – This property is applicable to serial sensors only. The default value is COM1 (that is, serial port 1). If you did not attach the sensor to serial port 1, change this property value as appropriate (COM2 for serial port 2, COM3 for serial port 3, and COM4 for serial port 4).
- *Unit* – This property is applicable only to those sensors that can have multiple receivers. These are InsideTRAK, Ascension Mouse, Bird, FASTRAK, and ISOTRAK II. The default value is 1, indicating that this object will represent the first receiver for the sensor. Change this value only if you are creating an additional receiver.

5 In the Nodes pane, create your Sensor object by dragging it to the Scene Graph pane.

6 Link the sensor to a Viewpoint or Movable object by creating a motion link (see "Motion Links" on page 144 for instructions).

Note You can link multiple Sensor objects to a particular Viewpoint or Movable object.

7 If desired, create additional receivers for the sensor (only available for InsideTRAK, Ascension Mouse, Bird, FASTRAK, and ISOTRAK II). To do so, increment the Unit property value (for example, set it to 2 to create the second receiver, 3 to create the third receiver, and so on). Then, create a new object for that receiver. Repeat this process to create the number of receivers that you want.

Note You can delete a Sensor object just like any other object (see "Deleting an Object" on page 88). When you delete a Sensor object, any motion link that references the deleted Sensor object will automatically be deleted.

Sensor Lag and Frame-Rate

WorldUp is designed so you can interact with computer-generated graphics flexibly and in real-time. Sensor objects provide a means of accomplishing this by directly coupling the user of an application to the geometry in the virtual world. The effectiveness of this interaction depends on several factors:

- *Sensor lag* – The time from when the sensor's state in the real world changes to when the sensor generates a record corresponding to that state; inversely proportional to sensor speed.
- *Sensor accuracy* – The range of values that a sensor may return when in a given state. This is usually specified as something such as ±0.1 inches within a range of 8 feet.
- *Frame-rate* – The number of frames per second that the system displays.

Note Even if your application runs with a high frame-rate, if the sensor lag is very large, then the user's impression of being able to interact in the virtual world may suffer. For very precise manipulations within the virtual world, the shorter the lag time, the better the user control.

Working With a Specific Sensor

This section describes each sensor supported by WorldUp, including any special configuration settings (such as, DIP switch settings, display settings, etc.) and a list of defined constants for any sensors that have buttons or switches. You can use these defined constants in scripts to refer to the sensor's buttons/switches.

Note Not all sensors have special configuration settings. The configuration procedures described may not be the only way to hook up a particular sensor.

For more detailed information on hooking up a particular sensor to your computer, refer to the sensor manufacturer's documentation or consult Sense8's web site at:

http://www.sense8.com/sensorsetup/index.html

Note When DIP switch settings are shown, the value 1 refers to ON, the value 0 refers to OFF. Sensors may vary as to whether ON is the up or down position for the DIP switch. The only exception to this is the InsideTRAK in which 1 refers to OFF whereas 0 refers to ON. Also, for serial sensors, the DIP switches corresponding to the baud rate are set to the value that WorldUp uses as default. So if you want to use a different baud rate, refer to the documentation for that sensor to change the DIP switch settings to the desired baud rate.

In the Type Workview, the Sensor object type contains subtypes for each sensor that WorldUp supports. Stand-alone sensors are located directly

within the Sensor type. Serial sensors (sensors that need to be plugged into the serial port) are located within the Serial Sensor subtype.

Mouse

The Mouse object type represents a standard mouse input device. By default, WorldUp provides you with a Mouse object named The Mouse. This object exists so you can use your mouse to navigate through worlds and select and move objects while creating a simulation.

Note You can have only one Mouse object. Since WorldUp creates a Mouse object automatically when it starts, you cannot create additional Mouse objects.

Defined Constants

The following constants define the event generated each frame that the button moves from up to down.

- LEFTDOWN
- MIDDLEDOWN
- RIGHTDOWN

The following constants define the event generated each frame that the button is held down.

- LEFTHELD
- MIDDLEHELD
- RIGHTHELD

The following constants define the event generated each frame that the button moves from down to up.

- LEFTUP
- MIDDLEUP
- RIGHTHUP

These can be used in scripts to access button press events as shown in the following example.

Sample Script Using Defined Constants

```
Sub Task ( m as Mouse
If m.MiscData and RIGHTHELD then
 message "The right mouse button is _
   depressed"
ElseIf m.MiscData and MIDDLEHELD then
 message "The middle mouse button is _
   depressed"
ElseIf m.MiscData and LEFTHELD then
 message "the left mouse button is _
   depressed"
End If
End Sub
```

Note The constants for the Mouse middle button are used only on three-button mice.

Ascension Mouse

The Ascension Mouse object type represents an Ascension Technology Corporation mouse, a six degree-of-freedom sensor that not only measures absolute position and orientation by using an electromagnetic field tracking system, but also has three user-programmable buttons like a standard mouse input device.

Defined Constants

The following constants define the event generated each frame that the button moves from up to down.

- ASCMOUSE_LEFTDOWN
- ASCMOUSE_MIDDLEDOWN
- ASCMOUSE_RIGHTDOWN

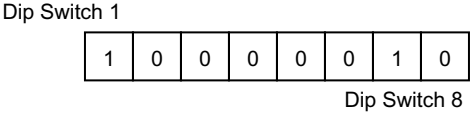
These can be used in scripts to access button press events as shown in the example for Mouse defined constant.

Ascension Bird

The Bird object type represents an Ascension Technology Corporation Bird, a six degree-of-freedom sensor that measures absolute position and orientation by using an electromagnetic field tracking system. Multiple bird units, known as Flock of Birds, are also supported.

Configuring the Bird

➤ To configure one Bird, set the DIP switches as shown below.



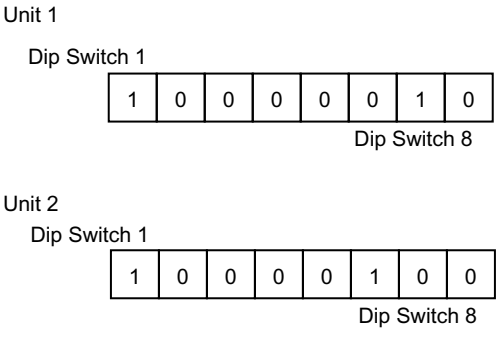
The DIP switch settings reference the following settings:

- Baud rate set to `19200` (switches 1, 2, and 3)
- Unit address set to `1` (switches 4, 5, 6 and 7)

Note The Bird must be set as unit 1, not 0.

- Mode set to fly (Switch 8)

➤ To configure two Birds, set the DIP switches as shown below.



Multiple Birds can be daisy-chained to make the Flock of Birds. See the Ascension Bird User's Guide for setup guidelines.

Note A flock can have up to 20 Bird units.

5DT Glove

The Glove5DT object type represents a Fifth Dimension Technologies' 5DT Glove, a data glove that measures finger flexure and the orientation (±128 degrees of pitch and roll with 0 being straight up) of a user's hand.

Gameport Joystick

The Gameport Joystick object type supports standard Gameport (analog) joysticks.

Installing the Joystick Driver for NT

You must have the Windows NT system driver for the gameport joystick installed to use the joystick with WorldUp. If you have not previously installed the driver or are unsure, follow the steps below:

To add the driver to your system

1 Insert the Windows NT 4.0 CD into your drive.

2 Open the Control Panel.

3 Select Multimedia.

4 Select Devices.

5 Select Add.

6. Choose Unlisted or updated driver, and press OK.

7. Type in `d:\drvlib\multimed\joystick\x86`, where `d` is the letter of your CD-ROM drive.

8. Restart system when prompted.

Calibrating the Gameport Joystick

WorldUp uses the standard Windows NT joystick control panel to calibrate the gameport joystick. You must calibrate before you use a joystick for the first time, and any time your joystick is not behaving correctly.

To calibrate your joystick

1. Open the Control Panel.
2. Select joystick.
3. Select the attributes that reflect your joystick.
4. Choose Calibrate and follow directions.
5. Choose Test,to verify your calibration.

Note Currently you can have only one joystick attached to your system. It must be on Port 1. (This is a limitation of the current Windows NT joystick driver).

Logitech 3D Mouse (Red Baron)

The Logitech 3D Mouse object type represents a Logitech, Inc. 3D Mouse, a desk-based sensor that measures absolute position and orientation by using three microphones to triangulate on three ultrasonic speakers. The speakers are mounted in a large triangle, and the microphones are in a smaller triangle, which is attached to the end of the mouse.

Note The Logitech 3D Mouse is also known as the Red Baron.

Defined Constants

The 3D Mouse has three buttons (left, middle, and right), similar to a normal Mouse. In addition, it has a button on the side of the mouse body called the *suspend button*, so named because it is used to suspend motion. When depressed, position and orientation records for the sensor are frozen at their current values, until the button is released. In this manner, the button can be used as a “clutch” or “ratchet” to be able to traverse large distances or angles by depressing the button while returning the sensor to within range of the ultrasonic speakers.

The following constants define the event generated each frame that the button is held down.

- LOGI_LEFTHELD
- LOGI_MIDDLEHELD
- LOGI_RIGHTHELD
- LOGI_SUSPEND

The following constant can be used to detect when the 3D Mouse is currently off the desktop. Note that this constant makes sense only if the 3D Mouse was on the desk at initialization.

- LOGI_FLYINGHELD

These can be used in scripts to access sensor data as shown in the example for Mouse defined constants on page 157.

Logitech Head Tracker

The Logitech Tracker object type represents a Logitech, Inc. Head Tracker, a head tracker sensor that measures absolute position and orientation by using three microphones to triangulate on three ultrasonic speakers. The speakers are mounted in a large triangle, and the microphones are in a smaller triangle, which is attached to the top of head-mounted displays for use as a head tracker.

Logitech Space Contol Mouse (Magellan)

The Magellan object type represents a Logitech, Inc. Space Control Mouse, a six degree-of-freedom table-top sensor that measures relative position and orientation by mapping the force and torque inputs applied to it.

Defined Constants

There are nine user-programmable buttons on the Space Control Mouse. All of these are positioned on the top edge of the Space Control Mouse frame. The button marked with an asterisk (*) is called the *pick button* (to maintain compatibility with the Spaceball).

The following constants define the event generated each frame that the button is held down.

- MAG_BTN1DOWN
- MAG_BTN2DOWN
- MAG_BTN3DOWN
- MAG_BTN4DOWN
- MAG_BTN5DOWN
- MAG_BTN6DOWN
- MAG_BTN7DOWN
- MAG_BTN8DOWN
- MAG_BTNADOWN

These can be used in scripts to access button press events as shown in the example for Mouse defined constants on page 157.

Polhemus FASTRAK

The Fastrak object type represents a Polhemus FASTRAK, a six degree-of-freedom sensor that measures absolute position and orientation by using an electromagnetic field tracking system. It can support multiple receivers (up to four) and has a much smaller sensor lag.

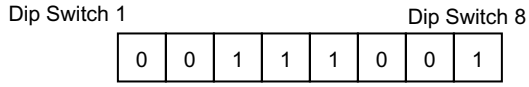
Configuring the FASTRAK

Note You must use a NULL MODEM cable to attach the FASTRAK to a PC.

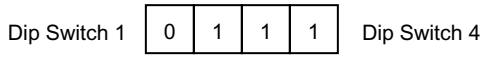
To configure one receiver

- Set the DIP switches as shown below.

Back



Front



The back DIP switch settings reference the following settings:

- Baud rate set to 19200 (switches 1, 2, and 3)
- No hardware handshaking functionality (switch 4)
- Character width set to 8 bits (switch 5)
- No parity (switches 6 and 7)
- RS-232 serial operation (switch 8)

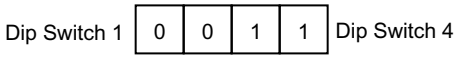
The front DIP Switch settings indicate how many receivers the FASTRAK currently has. The above setting indicates the FASTRAK has one receiver.

Note When a receiver is being used, its corresponding DIP switch is set to OFF.

To configure two receivers

- The DIP Switch settings in the back of the unit are the same as shown above for one receiver. Set the front DIP Switch settings as follows:

Front



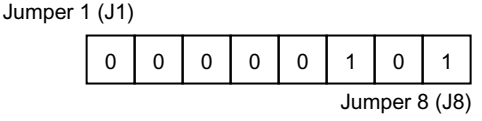
Note The FASTRAK can have up to four receivers.

Polhemus InsideTRAK

The InsideTrak object type represents a Polhemus InsideTRAK, a six degree-of-freedom sensor that measures absolute position and orientation by using an electromagnetic field tracking system. It is similar to the Polhemus ISOTRAKII, but is only available on Intel-based workstations with ISA bus slots.

To configure the InsideTRAK

1 Set the jumper switches to address 0x280 as shown below. The default setting is 0x300 but this is commonly used by network cards.



Note For the InsideTRAK, value 0 is ON and value 1 is OFF.

2 Copy the `ITRAKNT.SYS` file (the actual port driver for Windows NT) from the `\WORLDUP\DRIVERS` directory to your `\WINNT35\SYSTEM32\DRIVERS` directory (`\WINNT\SYSTEM32\DRIVERS` for NT 4.0 systems).

Note The remaining steps tell you how to configure the NT Registry for use with the Polhemus InsideTRAK. If you have WorldToolKit on your computer, you can skip the remaining steps since the WTK installation does this automatically.

3 Configure the NT Registry for use with the Polhemus InsideTRAK using the steps below. ( )

4 Double-click the file `\WINNT35\SYSTEM32\REGEDT32.EXE` (`\WINNT\SYSTEM32\REGEDT32.EXE` for NT 4.0 systems).

Note Do *not* use \WINNT35\REGEDIT.EXE (\WINNT\REGEDIT.EXE for NT 4.0 systems) for this configuration.

5 Click the "HKEY_LOCAL_MACHINE on Local Machine" registry window.

6 Navigate through the Registry hierarchy to the Services key: SYSTEM > CurrentControlSet > Services.

7 Click the Services key, then on the Edit menu, click Add Key.

8 In the Add Key dialog box, type `Itraknt` in the Key Name box and press ENTER.

9 Under Services, click the new Itraknt key.

10 On the Edit menu, click Add Value.

11 In the Add Value dialog box, type `Type` in the Value Name box and set the Data Type to REG_DWORD. Press ENTER.

12 In the DWORD Editor dialog box, type `1` in the Data box and press ENTER.

Note The Registry will show this value as `0x1`. This is normal.

13 Continue using the Add Value command on the Edit menu to add the following values to the Itraknt key:

Value Name	Settings
`Start`	Data Type: `REG_DWORD` Data: 2

Value Name	Settings
`Group`	Data Type: `REG_SZ` String: `Extended Base`
`ErrorControl` (no spaces)	Data Type: `REG_DWORD` Data: `1`

14 On the Edit menu, click Add Key.

15 In the Add Key dialog box, type `Parameters` in the Key Name box and press ENTER.

16 Under Itraknt, click the new Parameters key.

17 On the Edit menu, click Add Value and add the following values to the Parameters key:

Value Name	Settings
`IoPortAddress` (no spaces)	Data Type: `REG_DWORD` Data: `280`
`IoPortCount` (no spaces)	Data Type: `REG_DWORD` Data: `4`

18 Click Exit to end the session.

Note The InsideTRAK does not use interrupts, so the interrupt jumper should be set to disable interrupts.

Polhemus ISOTRAK/ISOTRAK II

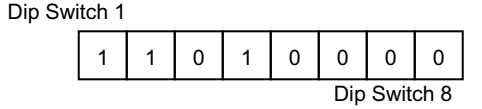
The Isotrak object type represents a Polhemus ISOTRAK, a six degree-of-freedom sensor that measures absolute position and orientation by using an electromagnetic field tracking system.

The IsotrakII object type represents a Polhemus ISOTRAK II, a two-receiver six degree-of-freedom sensor that measures absolute position and orientation by using an electromagnetic field tracking system.

Configuring the ISOTRAK and ISOTRAK II

Note You must use a NULL MODEM cable to attach the ISOTRAK or the ISOTRAKII to a PC.

➤ Set the DIP switches as shown below.



The DIP Switch settings reference the following settings:

- Baud rate set to 9600 (switches 1, 2, and 3)
- Internal Sync Mode (switch 4)
- Sync Generator OFF (switches 5 and 6)

Switches 7 and 8 are *don't care*.

Precision Navigation Wayfinder-VR

The CyberTrack object type represents a Precision Navigation Wayfinder-VR, a head tracker that measures absolute orientation using inertial and compass technologies. This tracker provides 360 degrees of yaw rotation, and about +/- 45 degrees of pitch and roll rotation.

Spacetec IMC Spaceball

The Spaceball object type represents a Spacetec IMC Spaceball, a six degree of freedom table-top sensor that measures relative position and orientation by mapping the force and torque inputs applied to it.

Defined Constants

There are nine user-programmable buttons on the Spaceball. Eight of these are positioned on the top edge of the Spaceball frame. One other button, called the *pick button*, is mounted on the forward face of the ball itself.

The following constants define the event generated each frame that the button moves from up to down.

- SBALL_BTN1DOWN
- SBALL_BTN2DOWN
- SBALL_BTN3DOWN
- SBALL_BTN4DOWN
- SBALL_BTN5DOWN
- SBALL_BTN6DOWN
- SBALL_BTN7DOWN
- SBALL_BTN8DOWN
- SBALL_PICKDOWN

The following constants define the event generated each frame that the button is held down.

- SBALL_BTN1HELD
- SBALL_BTN2HELD
- SBALL_BTN3HELD
- SBALL_BTN4HELD
- SBALL_BTN5HELD
- SBALL_BTN6HELD
- SBALL_BTN7HELD
- SBALL_BTN8HELD
- SBALL_PICKHELD

These can be used in scripts to access button press events as shown in the example for Mouse defined constants on page 157.

Note If you are using the Spacetec Spaceball Model 3003, be aware that the WorldUp driver has not been rewritten for the Model 3003. The 2003 driver works for both units, with a couple of differences. The ball controls translation and rotation in 6 degrees in real time for both models, but the 3003 only has one button (whereas the 2003 has 8 plus a pick button) that WorldUp supports. The button on the right side of the 3003 acts as the pick button.

StereoGraphics CrystalEyes and CrystalEyes VR LCD Shutter Glasses

The CrystalEyes VR object type represents a StereoGraphics, Inc. CrystalEyes VR stereo display and head tracker system that measures absolute position and orientation.

To configure the CrystalEyes VR

1 Setup the Emitter (EPC) unit. Attach it between the video out connector on your video card and your monitor using VGA cables.

2 If head tracking is required, connect the CrystalEyesVR to the Logitech unit and plug the serial cable from the Logitech unit into the serial port of your computer.

3 The next step depends on whether you want to configure the CrystalEyes VR for stereo or mono operation.

For *stereo* operation with an Intergraph GLZ graphics accelerator board, do the following:

- In Control Panel, set your screen display to stereo. This will reset your graphics card to output 120 Hz. You can set the resolution to whatever you desire, but be sure that your monitor can handle the 120 Hz at that resolution. To be safe, try it first at 640 x 480

(a 21" Intergraph can be set to 1024 x 768 @ 120 Hz) and increase the resolution as your monitor allows. Check with your monitor's specifications before you try. You will need to reboot your computer if you are running Windows NT 3.51. Windows NT 4.0 will reset your resolution, but the Intergraph driver must have a reboot to change to stereo mode.

For *mono* operation with non-GLZ graphics accelerator boards (ie: Stealth, Millenia, Mach, etc.), do the following:

- In Control Panel, set the display resolution to 640 x 480 @ 60 Hz. You will need to reboot if you are running Windows NT 3.51. Windows NT 4.0 will reset your resolution immediately if your video board supports it.

4 The emitter unit should be switched into either the high or low position (which controls signal range - usually set it to high).

5 In WorldUp, create a new Window (as described in "Windows, Viewports, and Viewpoints" on page 95).

Note Be sure to assign the correct Viewpoint object to the window.

6 Set the Width and Height properties for the Window object as appropriate (usually the same resolution as the screen.)

7 Set the Stereo property to True.

8 Set the Interleaved property as appropriate:

- *True* – for GLZ boards.
- *False* – for non-GLZ boards.

9 For stereo operaton, run the simulation and adjust the Parallax property of the Viewpoint object that is being used by the window to achieve maximum stereo effect (a parallax of 0.7 often works well).

ThrustMaster Formula T2 Steering Console

The FormulaT2 object type represents a Thrustmaster Technologies, Inc. Formula T2, a steering console that provides a natural driving experience around your virtual world.

To configuring the Formula T2

To configure the Formula T2, you must make the port driver available to your system, configure the NT Registry, and then calibrate the Formula T2.

To make the port driver available to your system:

➤ Copy the FORMULA.SYS file (the actual port driver for Windows NT) from the \WORLDUP\DRIVERS directory to your \WINNT35\SYSTEM32DRIVERS directory.

To configure the NT Registry:

Note If you have WorldToolKit on your computer, you can skip this section since the WTK installation does this automatically. Otherwise follow the instructions below.

1 Double-click the file \WINNT35\SYSTEM32\REGEDT32.EXE.

Note Do *not* use \WINNT35\REGEDIT.EXE for this configuration.

2 Click the HKEY_LOCAL_MACHINE on Local Machine registry window.

3 Navigate through the Registry hierarchy to the Services key: SYSTEM > CurrentControlSet > Services.

4 Click the Services key, then on the Edit menu, click Add Key.

5 In the Add Key dialog box, type `Formula` in the Key Name box and press ENTER.

6 Under Services, click the new Formula key.

7 On the Edit menu, select Add Value.

8 In the Add Value dialog box, type `Type` in the Value Name box and set the Data Type to `REG_DWORD`. Press ENTER.

9 In the DWORD Editor dialog box, type `1` in the Data box and press ENTER.

Note The Registry will show this value as `0x1`. This is normal.

10 Continue using the Add Value command on the Edit menu to add the following values to the Formula key:

Value Name	Settings
`Start`	Data Type: `REG_DWORD` Data: `2`
`Group`	Data Type: `REG_SZ` String: `Extended Base`
`ErrorControl` (no spaces)	Data Type: `REG_DWORD` Data: `1`

11 On the Edit menu, select Add Key.

12 In the Add Key dialog box, type `Parameters` in the Key Name box and press ENTER.

13 Under Formula, click the new Parameters key.

14 On the Edit menu, click Add Value and add the following values to the Parameters key:

Value Name	Settings
`IoPortAddress` (no spaces)	Data Type: `REG_DWORD` Data: `201`
`IoPortCount` (no spaces)	Data Type: `REG_DWORD` Data: `1`

15 Windows NT Registry input is now complete. On the Registry menu, click Exit to end the session.

To calibrate the Formula T2

At initialization, WorldUp searches the current directory for the formula calibration file named FORMULA.CAL. You can copy the default calibration file from the \WORLDUP\DRIVERS\directory to the correct directory, or you can create a new calibration file. To do this:

- Run T2CAL.EXE from the \WORLDUP\DRIVERS directory, which will generate the calibration file – FORMULA.CAL. This should be put in the same directory as the WorldUp executable. The following text reflects both the sample calibration file that ships with WorldUp (located in the \WORLDUP\DRIVERS directory) and the default values used by WorldUp:

  ```
  11 24 4 20 21 4
  ```

  The six entries specify integer values for wheel center, wheel range, wheel drift, pedal center, pedal range, and pedal drift. If the calibration file is not found, the default values are used.

Defined Constants

The following constants define the event generated each frame that the two buttons move from up to down.

- FORMULA_BUTTON1
- FORMULA_BUTTON2

The following constants define the event generated each frame that the shift knob is moved up or down.

- FORMULA_SHIFTUP
- FORMULA_SHIFTDN

These can be used in scripts to access sensor data as shown in the example for Mouse defined constants on page 157.

ThrustMaster Serial Joystick

The Serial Joystick object type represents a Thrustmaster Technologies, Inc. Serial Joystick (Mark II Flight Control/Weapons Control system) that measures relative position and orientation.

Configuring the Serial Joystick

At initialization WorldUp searches the current directory for the joystick calibration file named JOYSTICK.CAL. The following text reflects both the sample calibration file that ships with WorldUp (located in the \WORLDUP\DRIVERS directory) and the default values used by WorldUp:

```
0.0 255.0 0.0 255.0 128.0 128.0
```

The six entries specify floating point values for minimum X, maximum X, minimum Y, maximum Y, center X and center Y, respectively. If the calibration file is not found, the default values are used. You can copy the default calibration file from the \WORLDUP\DRIVERS directory to the correct directory, however it is better to create a new calibration file since the calibration values might differ for individual joysticks.

To calibrate the joystick

1 Create the Serial Joystick object as described in "Creating Sensor Objects" on page 155.

2 With the Serial Joystick object selected, on the Object menu click Calibrate Joystick (you must have rendering on to do this).

The Calibrate Serial Joystick dialog box displays.

3 Follow the instructions provided.

Note Since the joystick is calibrated upon creation of the Serial Joystick object, the object that you created in Step 1was actually calibrated using the values of the last calibration file created, or the WorldUp default values. You must create a new Serial Joystick object to use the new calibration file.

4 In the Type pane, click the Serial Joystick object that you created in Step 1 and click the Delete Selected button.

5 Create a new Serial Joystick object as described on "Creating an Object" on page 87.

This object is calibrated using the new calibration file.

Note You can continue to use this same calibration file for any sensor object corresponding to the same joystick device. You only need to recreate the calibration file if you are creating a sensor object corresponding to a different joystick device.

Defined Constants

The ThrustMaster Mark II Flight Control System supports three momentary buttons in addition to the trigger and a hat switch. The Mark II Weapons Control System adds an additional six momentary switches as well as a three-position rocker switch.

- SERJOY_TRIGGERDOWN
- SERJOY_TOPDOWN
- SERJOY_SIDEDOWN
- SERJOY_BOTTOMDOWN
- SERJOY_HATRIGHT
- SERJOY_HATLEFT
- SERJOY_HATDOWN
- SERJOY_HATUP
- SERJOY_WCS1

- SERJOY_WCS2
- SERJOY_WCS3
- SERJOY_WCS4
- SERJOY_WCS5
- SERJOY_WCS6
- SERJOY_WCS7
- SERJOY_WCSUP
- SERJOY_WCSDOWN

These can be used in scripts to access sensor data as shown in the example for Mouse defined constants on page 157.

VictorMAxx Technologies' CyberMAxx2 HMD

The CyberMaxx2 object type represents a VictorMaxx Technologies' CyberMaxx2 HMD, a head tracker that measures absolute orientation using inertial and compass technologies. This tracker provides 360 degrees of yaw rotation, and about +/- 60 degrees of pitch and roll rotation.

To configure the CyberMaxx2

1 Run the VGA signal through the daisy chain cable supplied with the CyberMaxx2.

2 Plug in the serial cable from the CyberMaxx2 into the serial port of your computer, if head tracking is desired.

The next step depends on whether you want to configure the CyberMaxx2 for stereo or mono operation.

3 For *stereo* operation with an Intergraph GLZ graphics accelerator board, do the following:

- In Control Panel, set the Intergraph driver for 640 x 480 @ 60Hz and true color (if your computer supports it). You will need to reboot if you are running Windows NT 3.51. Adjust the CyberMaxx2 video switch (on the right front side of the unit) to F1 for best results.

4 For *mono* operation with non-GLZ graphics accelerator boards (ie: Stealth, Millenia, Mach, etc.), do the following:

- In Control Panel, set the display resolution to 640 x 480 @ 60 Hz and true color. (Reboot if you are running Windows NT 3.51).

5 In WorldUp, create a new Window (as described in "Creating a Window" on page 97).

Note Be sure to assign the correct Viewpoint object to the window.

6 Set the Width and Height properties for the Window object as appropriate (usually the same resolution as the screen, for example 640 x 480).

7 Set the Stereo property as appropriate:

- *True* – for stereo operation on GLZ boards.
- *False* – for mono operation on non-GLZ boards.

Note Stereo mode on non-GLZ boards is used for the CrystalEyesVR, and not the CyberMaxx2.

8 For stereo operaton, run the simulation and adjust the Parallax property of the Viewpoint object that is being used by the window to achieve maximum stereo effect (a parallax of 0.5 often works well).

Virtual i-O i-glasses!

The IGlasses object type represents a Virtual i-O Corporation i-glasses!, a monoscopic and stereo head tracker that measures absolute orientation

using inertial and compass technologies. This tracker provides 360 degrees of yaw rotation, and about +/- 60 degrees of pitch and roll rotation.

To configure i-Glasses!

1 Attach the VGA breakout box between the video out connector on your video card and your monitor using VGA cables.

2 Plug in the serial cable from the breakout box into the serial port of your computer if head tracking is desired.

The next step depends on whether you want to configure the i-Glasses! for stereo or mono operation.

3 For *stereo* operation with an Intergraph GLZ graphics accelerator board, do the following:

- In the Control Panel, set the Intergraph driver for 640 x 480 @ 60 Hz and true color (if your computer supports it). You will need to reboot if you are running Windows NT 3.51. Adjust the Virtual I/O switch (on the right front side of the IGlasses) to 3D2.

4 For *mono* operation with non-GLZ graphics accelerator boards (ie: Stealth, Millenia, Mach, etc.), do the following:

- In the Control Panel, set the Intergraph driver for 640 x 480 @ 60 Hz and true color. (Reboot if you are running Windows NT 3.51).

5 In WorldUp, create a new Window (as described in "Creating a Window" on page 97).

Note Be sure to assign the correct Viewpoint object to the window.

6 Set the Width and Height properties for the Window object as appropriate (usually the same resolution as the screen, for example 640 x 480).

7 Set the Stereo property as appropriate:

- *True* – for stereo operation on GLZ boards.
- *False* – for mono operation on non-GLZ boards.

Note Stereo mode on non-GLZ boards is used for the CrystalEyesVR, and not the i-Glasses!.

8 For stereo operaton, run the simulation and adjust the Parallax property of the Viewpoint object that is being used by the window to achieve maximum stereo effect (a parallax of 0.5 often works well).

Note To get stereo with the Virtual I/O i-Glasses!, you need a hardware accelerator that can do line-interleaved stereo full screen (that is, *not in a window*) over a single channel at 640 x 480 @ 60 Hz. Currently, the Intergraph GLZ boardset drivers do this. Most PC video cards are not capable of outputting line-interleaved stereo. Hence, the Virtual I/O i-Glasses! will be in monoscopic mode under most circumstances.

Working With the State of a Sensor

This section describes the common Sensor properties that define a sensor's state. These properties allow you to access or modify (writable properties only) the current state of a sensor. A sample of how to work with these properties from scripts is provided at the end of this section.

Note For information on properties that are specific to a particular sensor, search on the name of the object type from which you created the sensor in the online Help.

Sensor Properties

Not all Sensor properties are applicable to all sensors. For example, you cannot set the rotational speed of absolute position and orientation sensors, such as the FASTRAK or Bird. Thus, these sensors do not have the Angular Rate property.

The following table lists the Sensor properties applicable to each sensor. A detailed description of each property follows the table.

Sensor	Angular Rate (page 170)	Misc Data (page 170)	Rotation (page 170)	Sensitivity (page 170)	Translation (page 170)
FormulaT2	X	X	X	X	X
InsideTrak			X	X	X
Mouse	X	X	X	X	X
Ascension Mouse		X	X	X	X
Bird			X	X	X
CrystalEyesVR			X	X	X
CyberMaxx2			X		
CyberTrack			X		
Fastrak			X	X	X
Glove5DT			Pitch/Roll		
Gameport Joystick	X	X	X	X	
IGlasses			X		
Isotrak			X	X	X
IsotrakII			X	X	X
Logitech 3D Mouse		X	X	X	X
Logitech Tracker			X	X	X
Magellan	X	X	X	X	X
Serial Joystick	X	X	X	X	X
Spaceball	X	X	X	X	X

Angular Rate

This property is used to get or set the scale factor for a sensor's rotation records. The angular rate is the maximum rotation (in radians) around any axis that a sensor returns in any pass through the simulation loop. For example, suppose you have a Spaceball attached to a viewpoint. The Spaceball's angular rate determines the maximum rotation around any axis that your viewpoint rotates when you apply torque on the ball.

Not all sensors supported in WorldUp have their rotational records scaled in this manner. You cannot set the rotational speed of absolute position and orientation sensors, such as the FASTRAK or Bird. The default angular rate for all sensors is 0.087266 radians, or 5 degrees. It may be convenient to specify the angular rate in terms of the defined constant PI (for example, 45 degrees = PI/4).

Misc Data

This property is used to get an integer value in which miscellaneous data pertaining to the sensor, like button press events, are stored. This value, together with the defined constants, can be used in scripts to access sensor data (see the example for Mouse defined constants on page 157).

Rotation

This property is used to get the rotational value of the current rotation record. The rotational value is the combined effect of the Pitch (*x*-rotation), Roll (*z*-rotation), and Yaw (*y*-rotation). The rotation record is affected by the sensor's angular rate value (see Angular Rate above).

Sensitivity

This property is used to get or set the scale factor for a sensor's translation records. The sensitivity is the maximum magnitude of the translational input from the sensor along each axis (in the same distance units as the 3D geometry making up the virtual world) in any pass through the simulation loop. For example, suppose you have a Spaceball attached to a viewpoint. The Spaceball's sensitivity determines the maximum distance along each axis that your viewpoint moves when you push on the ball.

Not all sensors supported in WorldUp have their translational records scaled in this way. Thus, you cannot set the translational speed of some sensors, such as the CyberMaxx2 or IGlasses. Some of the sensors that are scaled in this way are the Spaceball, Magellan, and the Mouse. The default sensitivity value for all sensors is 1.0. Attempts to set a sensor's sensitivity to a negative value are rejected, with no change to the current sensitivity. See the example at the end of this section.

Translation

This property is used to get the translational value of the current translation record. The translation record is affected by the sensor's sensitivity value (see "Sensitivity").

Sample Script

The following example lets you access and change some of the properties affecting the state of the Spaceball sensor.

Note Some of the properties previously described are read-only. You can retrieve the values of read-only properties, but you cannot modify them.

```
Sub Task ( s as Spaceball )
Dim key as String
key = GetKey()
If key <> "" Then
  Select Case key
    Case "up"
```

```
        s.AngularRate =
        s.AngularRate*1.1
      Case "down"
        s.AngularRate =
        s.AngularRate*0.9
      Case "right"
        s.Sensitivity =
        s.Sensitivity*1.1
      Case "left"
        s.Sensitivity =
        s.Sensitivity*0.9
      Case "t"
        Dim trans as Vect3d

        s.GetTranslation trans
        Message "Current Translation"
        Vect3dPrint trans
      Case "r"
        Dim rot as Orientation

        s.GetRotation rot
         Message "Current Rotation"
        OriPrint rot
    End Select
  End If
  End Sub
```

18

Multi-User Simulations

The high-level networking functionality provided in WorldUp Release 5, provides you with the ability to easily develop multi-user 3D/VR networked applications for use over LANs or the Internet. The high level networking capabilities are designed to operate in conjunction with SENSE8's World2World server product.

If you have not purchased the World2World server product, you will not be able to take advantage of the high level networking capabilities described in this chapter to build multi-user simulations. See below for a brief description of World2World or contact SENSE8 for detailed information about the World2World product.

To allow multiple users to run and participate in the same simulation, each user (client) needs to be able to receive certain updates (changes in property values) made by the other participants. For example, suppose there is a graphical object in your simulation that you want each user to be able to manipulate. If one user drags the object to a new location, you will want the other users to also see that movement.

To achieve this, the affected property must be shared by both the client that is modifying the value and the clients that want to receive the new value. Each

change made to the value of a property is known as an *event*. When a property is shared, the events that are internally generated for each property value change are what allow the updated information to be automatically sent over the network to any other clients that have also shared that property. If desired, you can trigger reactions to occur in response to an event.

The mechanism by which property value changes are transmitted to all clients who are sharing the property is the World2World server product. The World2World server product consists of a Server Manager, Simulation Servers, and an optional Firewall Proxy. A multi-user client application connects to the Server Manager, which determines whether the client has the appropriate log-in authority and directs the client to the appropriate Simulation Server, based on the simulation that the client is running. The Simulation Server stores and organizes simulation data and distributes data updates as appropriate to other users of the multi-user application connected to the same Simulation Server.

In WorldUp, you specify how the simulation is to connect to the World2World servers, which object properties are to be shared, and how that shared data will be stored and organized on a World2World Simulation Server. By limiting network data transfer to only properties that have been shared, WorldUp and World2World help to reduce bandwidth usage.

WorldUp simulations can connect to multiple World2World Simulation Servers. Each connection made by WorldUp to a World2World Simulation Server is represented by a W2WConnection object. Each W2WConnection object is associated with one or more W2WSharedGroup objects, which are used to group together a set of shared properties. By default, when you create a W2WConnection object, a W2WSharedGroup object is automatically created (called `<connection name>Root`), and is associated with the connection. You can create additional sharegroups in a hierarchical arrangement under the default Root sharegroup.

Note For more information on the server-side components of World2World, including how to install, configure, and start the World2World servers, see the *World2World User's Guide*. This chapter discusses the client-side aspects of developing a multi-user World2World-compliant simulation.

Network Connections

When a client starts a World2World-compliant simulation, the simulation will connect to a server where the application's data is to be shared. This process begins with a login call to a World2World Server Manager at a specified port, which determines what simulation this client will be entering. The Server Manager then proceeds to direct the client to the Simulation Server that has been designated to host that particular simulation. Once connected to the Simulation Server, the client can begin creating sharegroups (see "Sharegroups" on page 184) and sharing properties (see "Shared Properties" on page 179).

To better understand how this process works, see the example provided in the *World2World User's Guide*, Chapter 4, "Starting and Ending World2World." As the developer of the simulation, you only need to worry about the host name of the machine on which the Server Manager is located and determining a unique, unused port on that machine that you can associate with your simulation. The system administrator will take care of configuring the World2World servers to ensure that the clients of your simulation are connected to the appropriate Simulation Server.

In WorldUp, connections are represented by the W2WConnection object type, located under W2WNetwork. Each W2WConnection object represents a unique connection to a specific port on the Server Manager.

As you develop your multi-user simulation you must enable networking for WorldUp before connections can be activated. Once networking is enabled, any W2WConnection objects whose Connect property is set to True will attempt to connect to their specified machines and ports.

If you are running the multi-user simulation from a WorldUp Player, networking is always enabled, and any W2WConnection objects whose Connect property is set to True will immediately attempt to connect to their specified machines and ports.

Connected Users

Once a client has connected to a Simulation Server, it will be assigned a user name and Id. If the UserName property for the W2WConnection object is blank, the computer's system-defined name will be used.

If system-defined names are undesirable, you can assign alternative names through the UserName property. Typically, you will set the UserName property through scripts. For example, you could start the simulation with the connection disabled (Enabled property is False), and prompt the user to type their name. You could then set the UserName property equal to the value entered by the user, and then enable the connection.

When a connection is active (see page 177), a W2WUser object is created for each user that is currently connected to any of the simulation's active connections. The Users property on a W2WConnection object lists which of those users are connected to that specific connection (that is, which users are connected to the same host name and port, specified by the W2WConnection object). You can also view this list in the Network Connection dialog box (see page 177).

Note You cannot create objects from the W2WUser object type. This type exists only to contain the objects that are automatically created upon receiving notification from the Simulation Server that a new user has been added to the simulation.

Update Rates

Connections have an update rate, which determines the maximum updates per second for the connection. This is the maximum number of times per second that the client will send data packets to the Simulation Server and the maximum number of times per second that the Simulation Server will send packets to the client. The lower the update rate, the lower the packet traffic over the network.

For example, suppose that changes to a particular property in the simulation are being queued every half second and the connection's update rate is once per second. Since the property changes are being queued more frequently than the connection sends them to the Simulation Server, the connection can compare the updates waiting in the queue and only forward the most recent update for each shared property.

Note See "Update Frequencies" on page 179 for information on how to set the update frequency for each shared property. It is important to consider these settings when choosing your connection's update rate.

For modems, or other low-bandwidth mediums, the connection's update rate should be as low as possible. The default behavior is for the connection to match the client frame-rate (the UpdateRate property is set to `0`).

Reducing the connection's update rate may require you to use dead-reckoning techniques to smooth the data updates on receiving clients. An example of dead reckoning is provided in the Samples\Network\Van_go (intermediate) directory which is in the directory where you installed WorldUp.

User Added and User Removed Events

Any time a user connects to or disconnects from a connection, the Simulation Server is responsible for sending notification of this change to any other client that is connected to that connection.

When a client is notified of the addition of a user, a corresponding W2WUser object is created on the client's machine which generates a User Added event. If a response has been specified for that event, it will be triggered. For example, you might want to execute a script that creates a graphical object representing the new user.

Likewise, when a client is notified of the removal of a user, the corresponding W2WUser object is deleted from the client's machine, which generates a User Removed event. If a response has been specified for that event, it will be triggered.

For more information on events, see the *WorldUp Programmer's Guide*.

Working With Connections

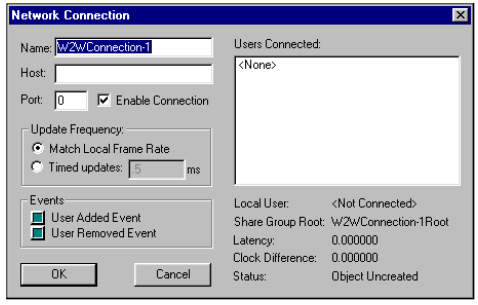
You can add, edit, and delete connections from the Type Workview or from the Network Browser. Once you have created a W2WConnection object and supplied the necessary parameters, you can activate the connection, which connects it to the World2World servers.

To create a new connection

1 On the Type Workview, select W2WConnection object located under W2WNetwork.

2 Click the Instantiate Selected Type button.

The Network Connection dialog box displays where you can set the connection's parameters and view statistics about the connection.



3 Set the desired parameters for the connection as described in "Network Connection Dialog Box" on page 177 and click OK.

Note If you are not sure how you want to set these parameters at this time, specify a name for the connection and set the remaining parameters later. While you will not be able to connect to the World2World servers without a specified host and port, having a W2WConnection object will allow you to begin creating sharegroups and sharing properties under that connection.

The new connection is added to the Network Browser, along with a default root sharegroup. Corresponding W2WConnection and W2WSharedGroup objects are also added to the Type Workview.

For more information on sharegroups, see "Sharegroups" on page 184.

To edit or view statistics about a connection

➤ Select Network Browser from the Networking menu.

The Network Connection dialog box displays from which you can modify any of the connection's parameters, and view the connection's statistics. Each component of the Network Connection dialog box is described in detail in the table on page 178.

To delete a connection

1 In the Type Workview, click the desired W2WConnection object located under W2WNetwork.

2 Click the Delete Selected  button.

To activate a connection

1 Start the Server Manager and Simulation Server as described in the *World2World User's Guide*.

2 Select the W2WConnection object in the Type Workview and set its Enabled property to True to ensure that the connection you want to activate is enabled.

3 In WorldUp, turn on networking mode in one of the following ways:

- On the Networking menu, select Enable Networking.
- In the Network Browser, check the option called Networking Enabled.

WorldUp attempts to connect all enabled connections in the simulation to the World2World servers.

Network Connection Dialog Box

As previously described, the Network Connection dialog box allows you to set a connection's parameters and view statistics about the connection. All of the information contained in the Network Connection dialog box is also reflected in the various properties of the W2WConnection object type.

To access the Network Connection dialog box, follow the steps for creating or editing connections in "Working With Connections" on page 176).

The following table describes each component of the Network Connection dialog box.

Network Connection Dialog Box Components

Dialog Component	Description
Name	The name of the connection.
Host	Host name or IP address of the machine to which you want to connect. This is the machine that you or your system administrator has designated as the host for the Server Manager.
Port	The number of the specific port on the host machine to which you want to connect. This is the port number that you or your system administrator has associated with this particular simulation. Note: When choosing a port number, keep in mind that ports 0 to 1024 are generally used by your operating system. You will probably want to specify a number between 1025 and 32,000. Check with your system administrator to determine whether certain ports are available.
Enable Connection	Indicates whether WorldUp will attempt to make a connection when Networking mode is enabled. The steps on activating a connection (page 177) describe how to enable Networking mode.
Match Local Frame Rate	Sets the connection's update frequency to the local machine's frame rate. See page 175 to understand update rates for connections.
Updates Per Sec	Allows you to specify the frequency (in seconds) for data value updates to be sent over the connection. See page 175 to understand update rates for connections.
User Added Event	Clicking the User Added Event button displays the Event Settings dialog box from which you can specify a response to occur each time a user is added to the connection. If you run a script as a response, WorldUp automatically inserts code after the entry point which determines whether the added user is the local user, as you will likely want to make use of this information. For information on how to specify event responses, see the *WorldUp Programmer's Guide*.
User Removed Event	Clicking the User Removed Event button displays the Event Settings dialog box from which you can specify a response to occur each time a user is removed from the connection. If you run a script as a response, WorldUp automatically inserts code after the entry point which determines whether the removed user is the local user, as you will likely want to make use of this information. For information on how to specify event responses, see the *WorldUp Programmer's Guide*.
Users Connected	When the connection has a status of Connected, this box lists all users who are also currently connected to that connection.
Local User	Displays the local client's name for this connection.
Share Group Root	The name of the root sharegroup for the connection.
Latency	Amount of time it takes for packets to be transmitted to or from a World2World Simulation Server.
Clock Difference	The time, in seconds, that the local and World2World Simulation Server clocks differ.
Status	The current status for the connection. See page 189 for a list of possible status messages and their meanings.

Shared Properties

As described in the introduction, when a client shares a property, the events that are internally generated each time that a client makes changes to the property's value cause those updates to be sent to the World2World Simulation Server. Once the update has been made on the Simulation Server, the Simulation Server sends the property update to all the other clients who are also sharing that property.

For more information about sharegroups, see "Sharegroups" on page 184.

Properties are shared under specific sharegroups. Each Simulation Server can have a hierarchical arrangement of sharegroups that are used to organize the properties stored on a Simulation Server. A single property can be shared under multiple sharegroups, though each Simulation Server will only retain a single copy of the shared property value.

When you share a property, internally a W2WSharedProperty object is created. The W2WSharedProperty object type is a hidden object type. That is, neither the type nor its objects appear in the Type Workview. However, through scripts, you can access and modify W2WSharedProperty objects just like any other object.

Locked Properties

Shared properties can be locked by a client, causing the Simulation Server to prohibit any other user from removing the property from its sharegroup or from modifying the property's value until the client (which holds the lock) releases the lock.

Only one client can have a lock on a particular property at any given time. However, any number of users can have active requests for a lock at the same time. When the client that currently owns the lock releases the lock, the lock will be passed on to one of the clients in the request queue.

Be aware that properties are also affected by locks on sharegroups in that a sharegroup lock trickles down to the sharegroup's properties (as well as its child sharegroups and their properties). See "Locked Sharegroups" on page 185 for information on locked sharegroups.

Persistent Properties

Shared properties can be flagged as being persistent. By making a shared property persistent, you ensure that the property will not be removed from the Simulation Server even if all of the clients who are sharing the property have disconnected from the Simulation Server. If a property is not persistent, the property will be automatically removed from the Simulation Server when there are no remaining clients who are sharing that property.

Note that a shared property will be persistent if at least one client who has asked to share the property has specified that the property is to be persistent.

Update Frequencies

Shared properties have an update frequency, specified in seconds, which determines how often property value updates are queued up to be sent over the network. It is the *connection's* update rate (see "Update Rates" on page 175) that controls how often the queued updates are actually sent across the network.

For details about this command, search on SendUpdate in the online Help.

The default for shared properties is to queue updates each time the property value changes. You can override this default and set the update frequency to queue updates at a set time interval, specified in seconds. For example, a value of 2 would send an update every two seconds. You can also set the update frequency to send no automatic updates. In this case, you would have to use scripts to call the SendUpdate command to manually update the property at the appropriate times.

Note that it is pointless (and inefficient) to queue property updates faster than they are actually being sent across the network. In fact, if you want to reduce network traffic, and you have shared properties whose update frequency is not critical to your simulation, you can queue property updates less often than they are being sent across the network, so that property updates aren't made more often than is really necessary. Take these factors into consideration when setting your properties' update frequencies and your connections' update rates.

Be aware that reducing the number of times that an update is sent across the network may require you to employ dead reckoning techniques to smooth the data updates on receiving clients. An example of dead reckoning is provided in the Samples\Network\Van_go (intermediate) directory of the directory in which you installed WorldUp.

Working With Shared Properties

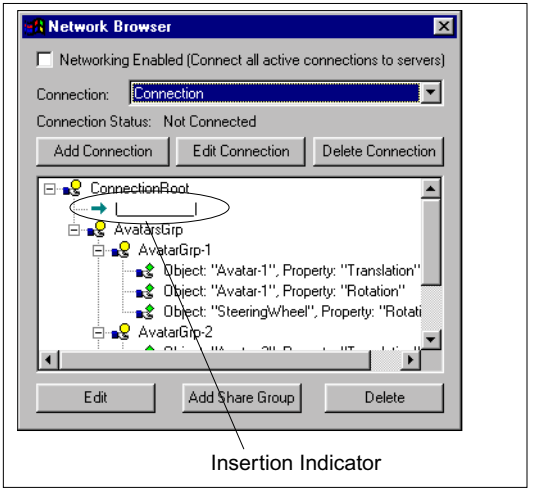
Before you can share a property, you must have at least one W2WConnection object in your universe. See page 176 for instructions. Note that you can share properties on objects only, not object types.

To share a property

1 Select Toggle Event Icons from the Networking menu to see the event icons in the Property pane. These icons give you instant feedback on which properties are shareable, as well as which are currently being shared.

2 In the Connection pull-down box, select the connection for which you want to share a property.

The Network Browser contains a data tree representing the connection's sharegroup hierarchy and the location of each existing shared property. The data tree also includes an insertion indicator, indicating which sharegroup will be used to contain new shared properties.



In the example above, any new shared properties would be contained directly within the ConnectionRoot sharegroup. You can position the insertion indicator under any sharegroup.

3 Click the insertion indicator and drag it under the sharegroup that you want to contain the property you are about to share.

4 In the Type Workview or Scene Graph pane, select the object that contains the property you want to share.

Note You cannot share properties on object *types*. Be sure to select a specific object.

5 In the Property pane, right-click on the property you want to share.

If the property is sharable, you can select the Event Settings option.

The Event Settings dialog box displays.

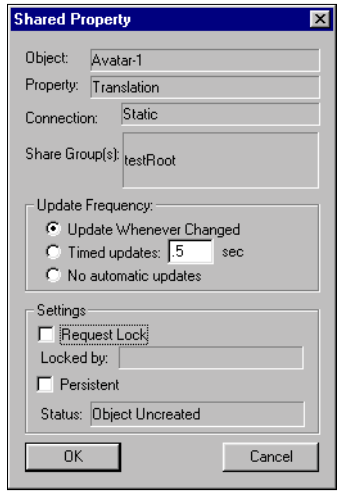
If the Event icon is grey, no events are generated for this property and you cannot access the Event Settings dialog box. For more information on events, see the *WorldUp Programmer's Guide*.

You can also right-mouse click on the property in the Property pane and click Share Property to bypass the Event Settings dialog box.

6 Check the option at the top called Share Property On Network.

If this option is not available, your project does not contain a W2WConnection object. See page 176 for instructions on creating one.

The Shared Property dialog box displays where you can set the shared property's parameters and view statistics about the shared property.



Notice that the Connection and Sharegroup values are set to the connection and sharegroup that you specified in the Network Browser.

7 Set the desired parameters for the shared property as described on "The Shared Property Dialog Box" on page 183 and click OK.

The shared property is added to the connection's data tree in the Network Browser.

Note Internally, a W2WSharedProperty object is created. W2WSharedProperty is a hidden object type. These objects do not display in the Type Workview, but you can manipulate them via scripts using WorldUp's functions and commands.

To edit a shared property

1 Open the Shared Property dialog box for an existing shared property by doing one of the following:

- On the Networking menu, select Network Browser. Select the desired connection. In the connection's data tree, double-click the shared property you want to edit.
- In the Type Workview or Scene Workview, select the object that contains the shared property you want to edit. In the Property pane, right-click the desired shared property. (The Event icon for a shared property has a lightning bolt over it.) In the Event Settings dialog box, click the check box next to the Share Property On Network option.

The Shared Property dialog box displays from which you can edit the shared property's parameters and view statistics about the shared property.

2 Set the desired parameters for the shared property as described on page 183 and click OK.

To move or copy a shared property to another sharegroup

1 On the Networking menu, select Network Browser.

2 Select the desired connection.

3 To move a shared property, in the connection's data tree, click the property and drag it onto the desired sharegroup.

4 To copy a shared property, press and hold the CTRL key while you drag the shared property onto the desired sharegroup.

The property is now shared under both sharegroups.

Note You cannot move or copy a shared property to another connection.

To unshare a property from the Event Settings dialog box

1 In the Type Workview or Scene Workview, select the object that contains the shared property you want to edit.

2 In the Property pane, right-click on the property you want to unshare and select Event Settings. (The Event icon for a shared property has a lightning bolt over it.)

3 In the Event Settings dialog box, uncheck the Share Property On Network option.


You can also right-click on the property in the Property pane and select Unshare Property to bypass the Event Settings dialog box.

The property becomes unshared and all instances of that property are removed from the connection's data tree in the Network Browser.

To remove or unshare a property from the Network Browser

1 On the Networking menu, select Network Browser.

2 In the Network Browser, select the connection that contains the property you want to unshare.

3 In the connection's data tree, click the property and click Delete.

If the property you deleted was the only instance of that property in the connection's data tree, the property becomes unshared. If there are other instances, you must delete them all in order to unshare the property.

The Shared Property Dialog Box

As previously described, the Shared Property dialog box allows you to set a shared property's parameters and view statistics about the shared property. Although W2WSharedProperty is a hidden object type, you can still access its properties through scripts. Search on W2WSharedProperty Type in the online Help.

To access the Shared Property dialog box, follow the steps for sharing properties (see page 180) or editing shared properties (see page 182).

The following table describes each component of the Shared Property dialog box.

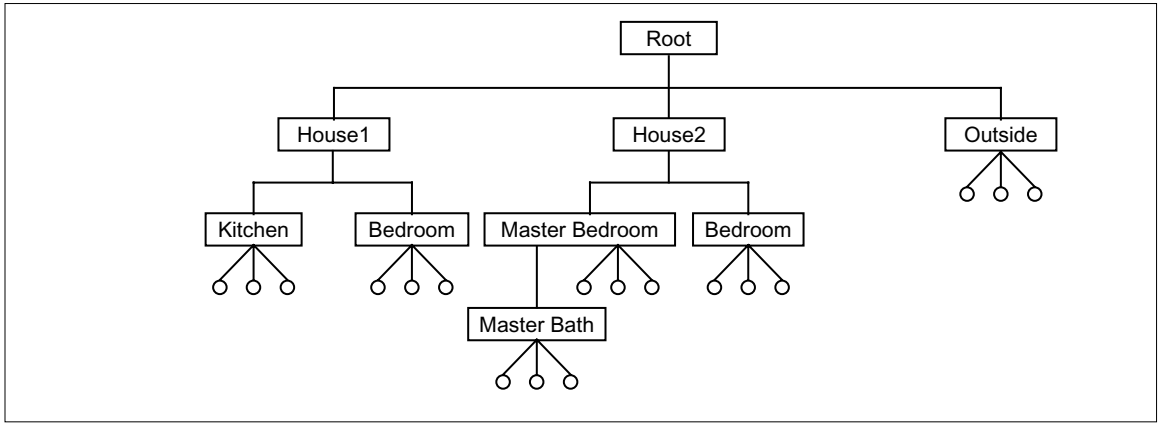
Dialog Component	Description
Object	The name of the object that contains the selected property.
Property	The name of the selected property.
Connection	The name of the connection for which this property will be shared. The connection that is indicated is the connection that was last selected in the Network Browser. So, if your universe contains multiple connections, ensure that the appropriate connection has focus in the Network Browser before sharing properties. Note: You cannot change this setting. To share a property under another connection, you must unshare the property, select the appropriate connection in the Network Browser, then share the property again. Also, you cannot share properties under multiple connections.
Share Group(s)	Lists the sharegroups under which the property is and/or will be shared. For information on how to share a property under multiple sharegroups, see page 182. The sharegroup under which the new share will be made is determined by the location of the insertion indicator in the Network Browser. See page 180 for details.
Update Frequency – The options in this section of the dialog box determine the frequency at which property change updates will be queued to be sent over the network. For more information on update frequencies for shared properties, see page 179.	
Update Whenever Changed	Indicates that a property change update will be queued every time the property is changed.
Updates Every	Indicates that a property change update will be queued at the indicated time interval, specified in seconds. For example, a value of 2 would send an update every two seconds.
No Automatic Updates	Indicates that property change updates will not be queued for the property and must therefore be sent manually with calls to SendUpdate. (Search on "SendUpdate" in the online help for information on this WorldUp function.)
Settings – The options in this section of the dialog box allow you to specify how the shared property will be treated.	

Dialog Component	Description
Request Lock	Requests a property lock for the local client so that other clients cannot modify this property. If the property has already been locked by another client, the name of the user who currently has the lock displays in the Locked By box. When that user releases the lock, the lock will be given to one of the clients in the request queue. For more information on shared property locks, see page 179.
Persistent	Flags the property as being persistent, ensuring that the property will not be removed from the Simulation Server. For more information on persistent properties, see page 179.
Status	The current status for the shared property. See page 189 for a list of possible status messages and their meanings.

Sharegroups

Sharegroups are container objects that are used to group one or more shared properties together on a World2World Simulation Server. Sharegroups can also contain child sharegroups. That is, they can have a parent/child relationship with other sharegroups so that a hierarchical arrangement of sharegroups can be created on a Simulation Server. Each connection has, by default, a root sharegroup. All other sharegroups created on that connection will be direct descendants (children) or indirect descendants of the root sharegroup. Sharegroups that are siblings (that is, they are children of a common parent sharegroup) must be uniquely named.

In the sample sharegroup data tree below, the Root, House1, and House2 sharegroups are placeholders. They contain no properties and exist only to add structure to the data tree.



Sample Sharegroup Data Tree

Locked Sharegroups

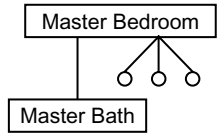
Sharegroups can be locked by a client, causing the Simulation Server to prohibit all other users from adding, moving, or removing properties or child sharegroups within the locked sharegroup's subtree, or from modifying the values of any properties contained within the locked sharegroup's subtree until the lock is released. That is, the lock on a sharegroup is recursive, affecting not only the properties located directly within the sharegroup, but also any of its child sharegroups and their properties. Locks are granted on a first-come, first-served basis.

In the preceding example, if a client placed a lock on the House1 sharegroup, its Kitchen and Bedroom child sharegroups, and all the properties contained within them would also be locked.

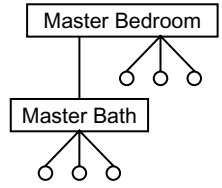
Registered Interest

While the Simulation Server keeps track of the data tree for all sharegroups and their properties, clients will only stay up-to-date with the sub-trees of sharegroups in which they have registered interest. Any time a property or sharegroup is added or removed from a sharegroup in the connection's data tree, the Simulation Server is responsible for sending notification of these changes to any client that is interested in that sharegroup.

In the sample data tree on page 184, suppose a client registers interest in only the MasterBedroom sharegroup. Upon registering interest, the client will immediately receive notification of the current children (sub-sharegroups) and properties of MasterBedroom.



Unlike locks, registered interest is not recursive. So, the client does not receive notification of the properties of MasterBath. To receive notification of the properties of MasterBath, the client would have to also register interest in MasterBath.



If any other client participating in the multi-user simulation adds or removes any sharegroups or properties to or from MasterBedroom or MasterBath, this client will be notified of the change so that it can stay up-to-date.

So, what happens when a client receives notification of an added sharegroup or shared property?

When a client is notified of the addition of a sharegroup, a corresponding W2WSharedGroup object is created on the client's machine, which generates a GroupAdded event. If a response has been specified for that event, it will be triggered.

For shared properties, it's a little more complicated. When a client is notified of the addition of a shared property, a corresponding W2WSharedProperty object is created (internally) on the client's machine, which generates a PropertyAdded event. If a response has been specified for that event, it will be

triggered. What happens next depends on whether the object and property being shared already exist in the client's simulation.

For example, if a client is notified that a property called Rotation of an object called Door was shared, and the Door object with a Rotation property already exists in the client's application, that property will be automatically shared.

If the Door object does not exist (or does not contain a Rotation property), and the PropertyAdded event does not trigger the execution of a script that creates the missing Door object and Rotation property, the W2WSharedProperty object that was internally created is deleted. If the PropertyAdded event does trigger the execution of a script that creates the Door object and Rotation property, the newly-created property will be automatically shared.

Note that registering and unregistering interest does not affect the distribution of shared property updates. Suppose a client registers interest in the MasterBedroom sharegroup, and (under the MasterBedroom sharegroup) shares the Rotation property of the Door object. If the client then unregisters interest in MasterBedroom, the client will still receive updates made to the Rotation property of Door, but will not be notified if another client removes or adds a property or sharegroup to MasterBedroom.

Persistent Sharegroups

Sharegroups, like properties, can be flagged as being persistent. By making a sharegroup persistent, you can ensure that the sharegroup and its properties will not be removed from the Simulation Server, even if all of the clients who are interested in the sharegroup or who are sharing one or more of the sharegroup's properties have disconnected from the Simulation Server. Making a sharegroup persistent has the same effect as making each of the sharegroup's properties persistent. Note, however, that making a sharegroup persistent does not affect the actual Persistent setting on each property.

A sharegroup will be persistent if at least one client has specified that the sharegroup is to be persistent.

When a persistent property or persistent sharegroup is hierarchically below a non-persistent sharegroup in the Simulation Server's data tree, the sharegroups that are ancestors of the persistent property or sharegroup are, for all intents and purposes, also persistent. The sharegroups from the root sharegroup down to the parent sharegroup of the persistent property (or sharegroup) must be retained on the Simulation Server to maintain the structural integrity of the sharegroups and properties stored within the Simulation Server.

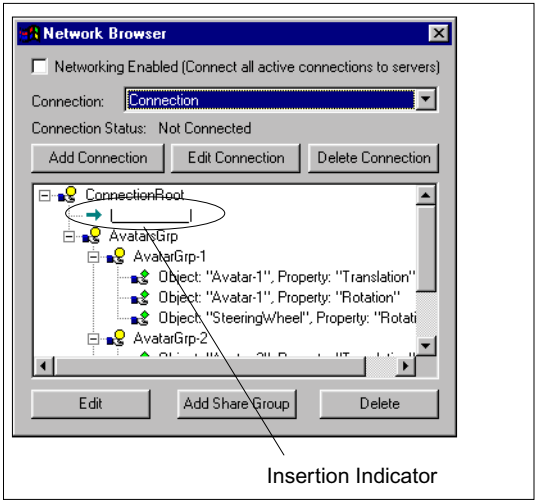
Working With Sharegroups

To create a sharegroup from the Network Browser

1 On the Networking menu, select Network Browser.

2 In the drop-down box at the top, select the connection to which you want to add the sharegroup.

In the middle of the Network Browser is a data tree representing the connection's sharegroup hierarchy and the location of each existing sharegroup and shared property. The data tree also includes an insertion indicator, which indicates under which sharegroup the new sharegroup will be added.

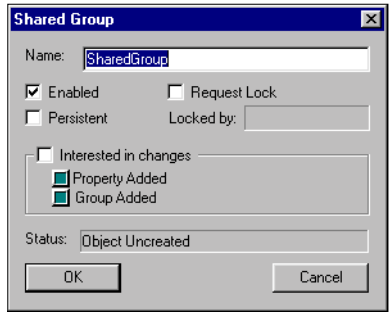


In the example above, any added sharegroups would be contained directly within the ConnectionRoot sharegroup. You can position the insertion indicator under any sharegroup.

3 In the Network Browser, click the insertion indicator and drag it under the sharegroup that you want to contain your new sharegroups.

4 Click the Add Sharegroup button.

The Shared Group dialog box displays from which you can set the sharegroup's parameters and view statistics about the sharegroup.



5 Set the desired parameters for the sharegroup as described in "The Shared Group Dialog Box" on page 188 and click OK.

A W2WSharedGroup object is created and added to the connection's data tree under the sharegroup where the insertion indicator was last positioned. page 188 describes how you can reposition the sharegroup in the data tree.

To create a sharegroup from the Type Workview

1 In the Type Workview, select W2WSharedGroup object located under W2WNetwork.

2 Click the Instantiate Selected Type button.

The Shared Group dialog box displays from which you can set the sharegroup's parameters and view statistics about the sharegroup.

3 Set the desired parameters for the sharegroup as described in "The Shared Group Dialog Box" on page 188 and click OK.

A W2WSharedGroup object is created and added to the connection's data tree under the sharegroup where the insertion indicator was last positioned.

To edit a sharegroup

1 Do one of the following to access the Shared Group dialog box:

- In the Network Browser, select the connection containing the sharegroup you want to edit, and double-click the sharegroup in the connection's data tree.
- In the Type Workview, double-click the desired W2WSharedGroup object.

The Shared Group dialog box displays from which you can edit the sharegroup's parameters and view statistics about the sharegroup.

2 Set the desired parameters for the sharegroup as described on page 188 and click OK.

To move a sharegroup

1 In the Network Browser, select the connection containing the sharegroup you want to move.

2 Click the sharegroup and drag it onto the sharegroup under which you want to position the selected sharegroup.

To delete a sharegroup

➤ Do one of the following:

- In the Network Browser, select the desired connection from the drop-down box at the top. In the connection's data tree, click the sharegroup you want to delete and click Delete.
- In the Type Workview, select the desired W2WSharedGroup object located under W2WNetwork.
- Click the Delete Selected  button.

The Shared Group Dialog Box

As previously described, the Shared Group dialog box allows you to set a sharegroup's parameters and view statistics about the sharegroup. All of the information contained in the Shared Group dialog box is also reflected in the various properties of the W2WSharedGroup object type.

The following table describes each component of the Shared Group dialog box.

Shared Group Dialog Box Components

Component	Description
Name	The name that you want to assign to the sharegroup.
Enabled	Indicates whether this sharegroup will be added to the data on the Simulation Server when the connection is active.
Persistent	Flags the sharegroup as being persistent, ensuring that the sharegroup (and its ancestors) are not removed from the Simulation Server. For more information on persistent properties, see page 179.
Request Lock	Requests a sharegroup lock for the local client so that other clients cannot add, move, or remove properties or child sharegroup's within the locked sharegroup's subtree, or modify the values of any properties contained within the locked sharegroup's subtree. If the sharegroup has already been locked by another client, the name of the user who currently has the lock displays in the Locked By box. When that user releases the lock, the lock will be given to one of the clients in the request queue. For more information on sharegroup locks, see page 185.

Component	Description
Interested In Changes	Allows you to register interest in the sharegroup. See page 185 to understand the effect of registering interest in sharegroups.
Property Added	Clicking the Property Added Event button displays the Event Settings dialog box from which you can specify a response to occur each time a W2WSharedProperty object is added to the sharegroup (see page 185 to better understand how this works). If you run a script as a reponse, WorldUp automatically inserts code after the entry point which determines whether the added property (and the object that contains the property) already exist in the client's application, as you will likely want to make use of this information. For information on how to specify event responses, see the WorldUp Programmer's Guide.
Group Added	Clicking the Group Added Event button displays the Event Settings dialog box from which you can specify a response to occur each time a W2WSharedGroup object is added to the sharegroup (see page 185 to better understand how this works). For information on how to specify event responses, see WorldUp Programmer's Guide.
Status	The current status for the sharegroup. See below for a list of possible status messages and their meanings.

Status Messages

Status messages are available for all connections, sharegroups, and shared properties.

You can access status messages as follows:

- *For connections* – open the Network Connection dialog box (see page 177), or from the Property pane access the Status property value for the desired W2WConnection object.
- *For sharegroups* – open the Shared Group dialog box (see page 177), or from the Property pane access the Status property value for the desired W2WSharedGroup object.
- *For shared properties* – open the Shared Property dialog box (see page 177), or through scripts access the Status property value for the desired W2WSharedProperty object. (W2WSharedProperty is a hidden object type, so you cannot access its properties from the Property pane.)

The rest of this section explains the various status messages that you may receive.

`Not Connected`

The object is not connected and is not trying to connect to the World2World servers. This could indicate one of the following:

- WorldUp is not in Networking mode (see page 177).
- For connections or sharegroups, the object's Enabled property is set to False.
- For sharegroups or shared properties, the object is not connected to a valid networking tree. In other words, the specified parent of the sharegroup or shared property is not a valid sharegroup for the connection.
- For sharegroups or shared properties, a parent object failed to connect.

`Attempting To Connect`
The object is trying to connect to the World2World servers.

`Connected`
The object is successfully connected to and communicating with the World2World servers.

`Failed To Connect`
The connection was attempted but was unsuccessful.

`Object Uncreated`
The object has not yet been constructed, or is in the process of being constructed.

`Conflict Encounter`
Two distinct objects with the same name have been registered for the same job.

For Shared Properties Only

`Property's Object Does Not Exist`
The W2WSharedProperty object refers to an object that never or no longer exists.

`Duplicate Share For Property`
Multiple W2WSharedProperty objects exist for the same property.

19

Tips and Tricks

This chapter provides performance hints, tips, and tricks, and responses to frequently asked questions.

Performance

When WorldUp is running, all my other applications are *extremely* sluggish.

WorldUp is designed to cheat every other application of processor time so that it can both run quickly and support its complex interface. When you want to use another application, you may find it necessary to turn rendering off before you switch to your other application. This forces WorldUp to release its stranglehold on the processor.

My simulation needs to run faster.

Use the Profiler (see "Running Simulations" on page 20) to track down what part of your simulation is taking the most time. Leave the Profiler open while moving to different locations or stages of your simulation, and note any changes in the performance profile. It may take you a while to understand the profiler information; give yourself some time to become familiar and comfortable with the statistics reported by the Profiler. The questions below address what you might be able to do in response to specific problems.

RayIntersect is taking all of my simulation time.

If used pervasively in your simulation, RayIntersect can begin to take quite a bit of time from your simulation. If you can give RayIntersect less nodes to search through, it will be faster. RayIntersect takes a starting node as an input parameter. If you pass in Root, it may have to search through far more than if you gave it a more targeted node, depending on your simulation. It's usually better to call it once for a lot of nodes, than many times on specific nodes.

Breaking down your scene such that the scene graph node hierarchy corresponds to the spatial layout of the scene will help RayItersect's performance considerably. This will give RayIntersect more information about the scene, and so it can be more efficient.

Collision detection is taking all of my simulation time.

There are many ways to minimize the number of collision detection calls being made. Do anything you can to isolate the objects you're interested in. You may want to iterate through all of the objects of a particular type and call IntersectMovable on each. This might succeed in just colliding against the objects you're interested in, but if you have a scene graph that incorporates smart spatial divisions, you might have more success by starting at a node and, if there's a collision, recursively iterating through the node's children and testing against each of them, until you've collided with a node you're interested in.

Creating objects from scripts is taking forever.

Creating objects from scripts is actually a reasonably fast operation. The slowness you're seeing is probably due to the fact that when an object is created in the development environment, WorldUp has to update the Workviews. This is slow. When the simulation is run through the WorldUp Players, you won't see this performance hit. If this drop in performance is bothering you, you can disable the Update Property Changes From Scripts option in WorldUp (see "Automatically Updating Properties" on page 92 for instructions).

Rendering is taking a long time.

In general, rendering is simply a very expensive process. If you don't have a graphics accelerator card, textures will cost dearly. If you have a good graphics accelerator card, textures may be no slower than non-textured surfaces; on these machines using low polygon, highly textured objects is a better trade off. There are more issues for creating efficient real-time 3D worlds than we have a forum to address in this manual. An obvious point is that the less polygons being rendered at once, the better. Use the Profiler to monitor how many polygons you're rendering.

Here are some general tips:

- To increase rendering speed in the Development window, turn off helper features, such as shadows, grids, and drop lines. You can do this by selecting the Display Options  button from the Development window toolbar.
- If a geometry is not going to be changing shape or appearance during the simulation, optimize it, using the Optimize property. It is best to do this when you're finished with your simulation, and not planning anymore scene alterations.
- One of the biggest performance problems is rendering parts of the scene that are obscured by closer objects. Try to minimize this type of useless rendering. For example, if you're in a room with all of the doors closed, you may not be able to see the rest of the house, but it may be being rendered. Try to disable areas that you can't see by using the Enabled property (note that if a parent node is disabled, all of its children

will also not be rendered). Also, you can decrease the value of the window's Yon Clipping property. This value represents the distance from the viewpoint at which objects will no longer be rendered.

- LevelOfDetail nodes are critical for making large simulations run reasonably. By creating geometries with several different versions of decreasing levels of complexity, LevelOfDetail nodes can manage swapping in less detailed versions when the object is far away.

  For more information on LevelOfDetail nodes, see "LevelOfDetail Nodes" on page 26.

- Using multiple lights can significantly hurt performance. If you need multiple lights, arrange them in the scene graph so they only affect the nodes that they absolutely need to.

How can I make my models more efficient?

Unless necessary for the simulation, models should be single-sided. That is, their polygons should only be visible from one side, the outside. Unnecessary use of doubled-sided polygons can skyrocket the cost of rendering a geometry.

Obviously, the fewer polygons, the better. Most professional modelers now have good optimizers. Make sure you use one of these optimizers, as many techniques used in modern modeling tools are very polygon intensive.

Rendering

Viewing Graphical Objects

If you don't see any objects in the Development window check the following:

- Is rendering turned on?

  If the Rendering On button is not pressed down, click it to toggle rendering on. Or, on the Options menu, click Rendering On if there is not already a check mark next to it.

- Does your simulation contain any graphical objects?

  In the Nodes pane of the Scene Workview, check to see that you have at least one object beneath one of the Geometry subtypes (Block, Cylinder, Imported, Sphere, or Text3d). If there are no graphical objects, create some as described in "Adding 3D Objects" on page 83.

- Are the graphical objects simply out of the window's current view, or is the viewpoint inside of your objects?

  Click the Zoom All  button or use other navigation techniques described in Chapter 8, *Development Window – Navigation and Manipulation*, to adjust your viewpoint so that you can see all of your graphical objects. Or, translate (move) the graphical objects away from the viewpoint using the methods described in "Manipulating Objects" on page 82.

  Note All Movable objects (except imported geometries) that you create are initially positioned at the center of the universe (0,0,0).

- Is the graphical object that you want to see inside of another graphical object?

  Translate (move) one of the graphical objects using the methods described in "Translating and Rotating Movables" on page 110. Or, move the viewpoint inside of the graphical object that is obstructing the view using the navigation techniques described in "Manipulating Objects" on page 82.

- If it's a small object that is close to the viewpoint, it could be that the value of your window's Hither Clipping property is too high and is excluding the object from rendering.
- If the object is large or far away, it could be that the value of your window's Yon Clipping property is too low and is excluding the object from rendering.

  Note Remember you have at least two windows: an Application window (Window-1) and a Development window (DevWindow-1). If you change the Hither Clipping or Yon Clipping values, be sure to make those changes to all appropriate windows.
- The Enabled property of the object or of one of the object's parents could be False.
- If it's an Imported object, there might be an invalid file name/entry name combination that doesn't point to a valid entry in a valid file. Select the Filename or Entry property in the Property pane, highlight the value in the Property text box, and press ENTER. This will trick the editor into thinking you just changed the property, and will try to reload the object. Check the Status window for an error message.

My rendering is wrong. My geometries get flashing, jaggy strips or bites cut out of them through which I can see objects or surfaces behind them.

It could be that the distance between your hither clipping plane and yon clipping plane are too far apart. The greater the difference between the values of your window's Hither Clipping and Yon Clipping properties, the less ability the renderer will have to distinguish which object is above the other. Try making these two planes closer together. Either that or move the objects farther apart.

It could be that you have two surfaces that are actually co-planar, that is, they sit exactly on top of each other. If this is true, the renderer will never be able to decide which surface is on top, and you will get the stripped effect. You will have to move the surfaces away from each other, far enough for the renderer to tell them apart.

My rendering is wrong. My textures, when viewed at an angle, get warped and distorted.

You probably have Texture Perspective turned off. If you don't have a graphics accelerator card, disabling this option can increase your performance significantly, if you're willing to suffer the warping textures. You can change this option in the Rendering Style dialog box (see "Setting Rendering Parameters" on page 41 for instructions).

Sounds

Sounds won't play.

There are a variety of reasons why sounds might not play:

- Does your sound system work?

  Make sure that your sound system is properly set up. Use the sound player that comes with your operating system to try and play a sound. If not, it could be that your sound drivers are not installed or are corrupted, or your sound card is not working, or your system volume is simply turned all the way down. Check your system's sound settings.
- Does the sound have its Playing property set to True?

  To play a sound, the Sound object's Playing property must be set to True. You can set this directly through the Property pane, or by calling

the script method Play. (The command would be `snd.Play`, where `snd` is a variable pointing to a sound.)

- Is something hogging the sound device?

  Another application might be running that's holding on to the sound device, not letting WorldUp play its sounds. Close that application.

- Is Audio: Listener set correctly?

  The Audio: Listener property of the Universe object should be pointing to the correct viewpoint (whichever viewpoint is used in the main application window).

- What is Audio: Rolloff set to?

  The Audio: Rolloff property of the Universe object determines how fast the sounds in space fall off. If the number is too low for the size of your universe, sounds might be falling off too quickly.

Sounds are not spatialized correctly.

- Is the Attached To property set to the correct object?

  Make sure the Sound object's Attached To property is set to a valid object, which is still in your scene.

- Is Audio: Listener set correctly?

  The Audio: Listener property of the Universe object should be pointing to the correct viewpoint (whichever viewpoint is used in the main application window).

- What is Audio: Rolloff set to?

  The Audio: Rolloff property of the Universe object determines how fast the sounds in space fall off. If the number is too high for the size of your universe, you might not, in the area in which you move in your universe, be able to tell the difference in roll-off. Try decreasing this number and see if the results are better.

My sounds is playing too fast or too slow.

If you are using the DiamondWare sound system (this is the default for WorldUp on the Windows platform – check the Audio Device property on the Universe object), sounds must be recorded at a frequency of 22 K. Any sound editor will allow you to resample a sound file.

Make sure the sound's Pitch property is set to 1.0.

Fonts

How do I create a 3D font?

A 3D font file can be of any supported 3D model type. You can create a new 3D font file by making a model of each character and/or symbol you want. Then you must name each of these objects with the char prefix followed with that character/symbol's ASCII value. For example, the letter a would be named char97 and A would be char65.

How do I change 2D fonts and their size?

First, you need to find out which fonts are installed on your machine. In Windows, you can do this by going to the Control Panel and choosing Fonts. This will display all the fonts currently installed. Now, create a file named FONT.WTK and put the name of the font you want followed by the size. This FONT.WTK file *must* be in the directory of the WorldUp executable you are running. If you are running a plug-in, the FONT.WTK file must be in your plug-in directory. The only problem here is, in order for someone else to view your font, they must also have the FONT.WTK in *their* executables

directory *and* they must have that font installed on their computer. Your best bet is to just go with the default font of WorldUp.

Miscellaneous

I change the Properties of an object, but the simulation doesn't change!

or

All the properties of the Universe are displayed in red text (implying read-only), how do I change them?

Make sure you are editing the object itself and not the object type. Changing the property values of the type will affect only newly created objects of that type, but not existing objects. For the Universe type, all of the properties have been made read-only. Since you can't create a Universe object (you can only have one universe, and it is created for you), there is no purpose in changing the type's properties.

My LevelOfDetail node isn't switching at the right time, or isn't switching at all.

1 Select the LevelOfDetail node and see where the axes are.

2 If you don't see any axes, click the Zoom To Selected  button in the Development window.

The axes represent the LevelOfDetail's center. This is where the distance from the LevelOfDetail is measured from. Move the LevelOfDetail node to where you want the center to be and make the translation of the LevelOfDetail children `0,0,0`, or whatever offset you want from this center.

When I rotate my object, it doesn't rotate around the center I want.

If you're rotating a geometry, you need to adjust the Origin Offset property. (See "Adjusting a Geometry's Pivot Point" on page 116 for instructions on how to do this in the Development window.)

If you're rotating a group, remember that the objects contained within that group will rotate around the Group node's pivot point, not their own. You will either need to move the Group node's pivot point as described above, or else move the entire location of the Group node, and then move the children back where you want them. A frequent mistake with Group nodes is to neglect to translate its children to the group's center (if that's what you want). An easy way to do this for groups with only one immediate child is to select the child object, copy its translation (right click on the Translation property and select Copy), then select the group, paste the translation in (by right clicking on the property and selecting Paste), and then go back to the child object and type `0,0,0` in for the translation).

This dumb bar appears when I load my world in a plug-in.

That is the navigation bar. By default, it is on in the plug-ins. To turn it off, put the following command in your Startup script:

```
NavBarOptions 0
```

I output my universe as a VRML file and it is upside down in my viewer. How do I fix this?

This is not actually a bug. It is just that WorldUp has its coordinate axis with the Y reversed from the VRML specifications. Rotate your viewpoint 180˚ in the Y (yaw) and save it out as a VRML file again.

How do I implement database connectivity?

You can implement database connectivity via BasicScript's SQL functions. The two SQL samples provided with WorldUp are located in the Samples\SQL subdirectory of the directory in which you installed WorldUp. For details on BasicScript's SQL functions, search on SQL in the online Help.

Model Tricks

How do I create versions of a model, varying in detail, to successfully work with LevelOfDetail nodes?

In creating different levels of detail, the process is to usually start with your most complex or highest level of detail geometry and to create copies of it with fewer and fewer polygons. The trick is to keep enough (and the right) polygons such that the general topology, color, and texture is maintained in the process and the switch from one geometry to another is visually seamless. With some simple geometric shapes (such as spheres, cones, cylinders) this is a trivial effort requiring you to generate multiple versions of the object.

For information on LevelOfDetail nodes, see "LevelOfDetail Nodes" on page 26.

The more typical problem, however, is where you attempt to create three or four versions of a house, car, or terrain. Various tools and techniques are available that can help you in this process, such as the following:

- *Automatic mesh simplifiers* – Several companies offer tools that will take a meshed surface and create reduced versions while maintaining topology and textures. They usually only work with meshed surfaces, which isn't always the case with many models.
- *Image processing program (texture replacement)* – At a suitable distance, a single texture or several crossed textures can replace a highly complex object. You can use WorldUp to render the complex object, save the image out to a file, and then use an image processing program to create a suitable texture that is then mapped onto several polygons crossed like an "X".
- *Modeling programs* – Most modelers allow you to manually reduce polygon count. A rare few, like Multigen, have some tools to help create LevelOfDetail nodes. In general, it takes experience to learn how to simplify enough – but not too much – to create proper LevelOfDetail nodes.

How can I create radiosity-preprocessed models?

As with Gouraud shading, use of vertex colors can be used to increase the visual realism of your virtual scene.

For example, vertex color support enables you to render models that have been radiosity-preprocessed. A radiosity-preprocessed model stores lighting information such as shadows and reflections as vertex colors – this lighting doesn't then have to be computed at run-time. The result is complex lighting with real-time performance.

A radiosity preprocessor is a program that takes as input a model and a light source specification and generates a new model with lighting information (such as for shadows or reflections) built into it. This involves meshing the original model to contain more detailed color information. This color information is stored at the vertices of the mesh, and WorldUp (or the hardware that WorldUp is running on) interpolates between these vertex color values to produce a smooth effect.

For a better understanding, take a look at any of the several radiosity rendered models on the WorldUp CD in the \models\radiosity directory.

In addition to storing lighting information, vertex colors can also represent other values such as the temperature or pressure throughout an object.

Vertex colors can be set for geometries in the following ways:

- In an NFF file. Search on NFF File Format in the online Help for instructions.
- Using a radiosity preprocessing program. ATMA's program called Real Light is a radiosity preprocessor that reads and writes NFF files.

20

Publishing Your Application

Once you have finished building your application, you need to determine how you will distribute it to other users. WorldUp provides you with three options:

- As a stand-alone application/simulation
- As an ActiveX control that can be embedded in another application
- As an internet plug-in

Whichever you choose, you need to ensure that the target system gets the project file (.UP) as well as all of the models, scripts, images, behaviors, DLLs, sounds, and whatever else is included in your simulation. You can do this yourself, or you can export your simulation using WorldUp's Simulation Packager, which packages all the components into a single file, encrypts your scripts, and compresses the whole package into a single ZUP file.

In addition to your simulation files, you need to consider whether the correct player is installed on the target computer. You should also consider whether you want to distribute source script files (.EBS) or encrypted script files (.EBX).

This chapter covers WorldUp's role in these key distribution choices.

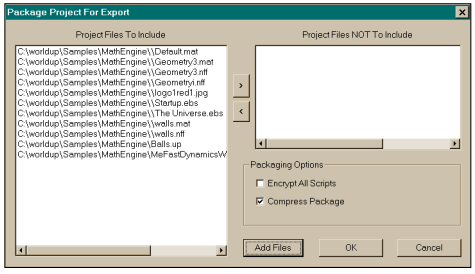
Packaging the Project for Distribution

The simplest mechanism for distributing your application is through WorldUp's Simulation Packager, which collects all the files WorldUp determines are necessary to your simulation.

To export your simulation as a WorldUp package file

1 Select Export Package As > WorldUp Package from the File menu.

The Package Project For Export dialog box appears.



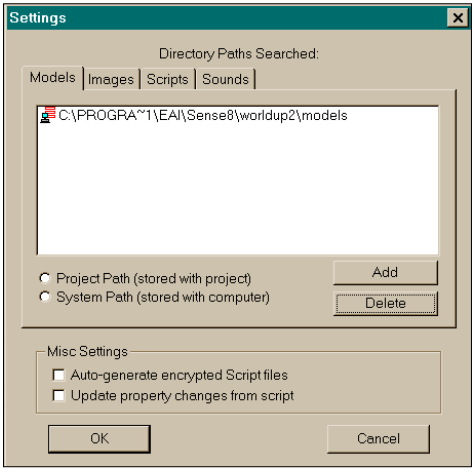
The Package Project for Export dialog box gives you the opportunity to:

- Add/Remove any additional files to the package.
- Encrypt your BasicScript script files to protect your source code from being copied.
- Compress the entire package.

The packager then places all the files into a single ZUP file with the same name as your project in the simulation directory. A ZUP file can be opened by either WorldUp or any of the WorldUp Players. When you open a ZUP file, WorldUp expands all the files contained in the ZUP file into a WuCache directory (this directory is in your Windows directory), and then proceeds to load the project file.

The following gives some important notes about ZUP files:

- In many situations, a file your simulation depends on is not automatically detected by WorldUp, including geometry files that get instantiated dynamically, external DLLs called from script, etc. You need to add these files manually using the Add Files button on the Package Project For Export dialog box.
- WorldUp only adds model files for currently instantiated objects. If you will be instantiating other models during your simulation, you need to manually add these models and their textures.
- When you open a ZUP file, WorldUp checks the WuCache directory to see if this file already exists. If it does, WorldUp opens the already expanded version; otherwise WorldUp expands the ZUP file into the WuCache directory. The danger occurs if you were then to create a new ZUP file of the same name. When you attempted to open the new file, WorldUp would open the existing expanded ZUP file, and you would not see any changes. To be safe, you should first clear the WuCache directory of the previous project files.
- As you've discovered from using WorldUp, it is extremely important that WorldUp know where models, textures, scripts, and sounds are located. WorldUp does this using directory paths, which you control in the Universe Settings dialog box.



Setting Project Paths within WorldUp

When planning to export a project manually, you should add the requisite directories to the Project path so that when your simulation is opened in one of the players, the UP file itself knows where to look for these files. By default, WorldUp looks in the same directory as the UP file, which is why ZUP files work, since it expands all files into one directory. If you are creating your own distribution mechanism, however, you will find this invaluable.

Choosing a Player

Three different WorldUp players can be used to distribute your simulation:

- **Stand-alone Player** – A stand-alone executable that loads and runs simulation files.
- **Internet Plug-in** – A DLL that loads and runs simulations inside of Netscape Navigator or Internet Explorer.
- **ActiveX Control** – An OCX that can embed your simulation inside of ActiveX Control Containers, such as Visual Basic, Internet Explorer, or Macromedia Director. Exposes an interface allowing communication between control and container.

Which one you choose depends upon your project's requirements. In addition, all players come in both an OpenGL and a Direct3D version. Before choosing your player, consider the following issues.

Is my simulation an immersive or desktop application?

If you are creating an immersive simulation, you want to create borderless windows for your application, and run it in the stand-alone player. This way, no user interface interference occurs with your simulation. If however, you are running your simulation as a desktop application, you may consider running it with a traditional interface, which means integrating it into a Visual C++ or Visual Basic application using the ActiveX Control.

Will my simulation run with other applications?

If the simulation works with other applications, consider embedding the simulation in another application.

Will my simulation be running in a Web Browser?

If your simulation is an Internet application, you can use the Internet plug-in or the ActiveX control. The ActiveX control only works in Internet Explorer, however it's ActiveX interface wields much more flexibility than the standard Netscape plug-in.

Will my simulation run on a desktop or a laptop?

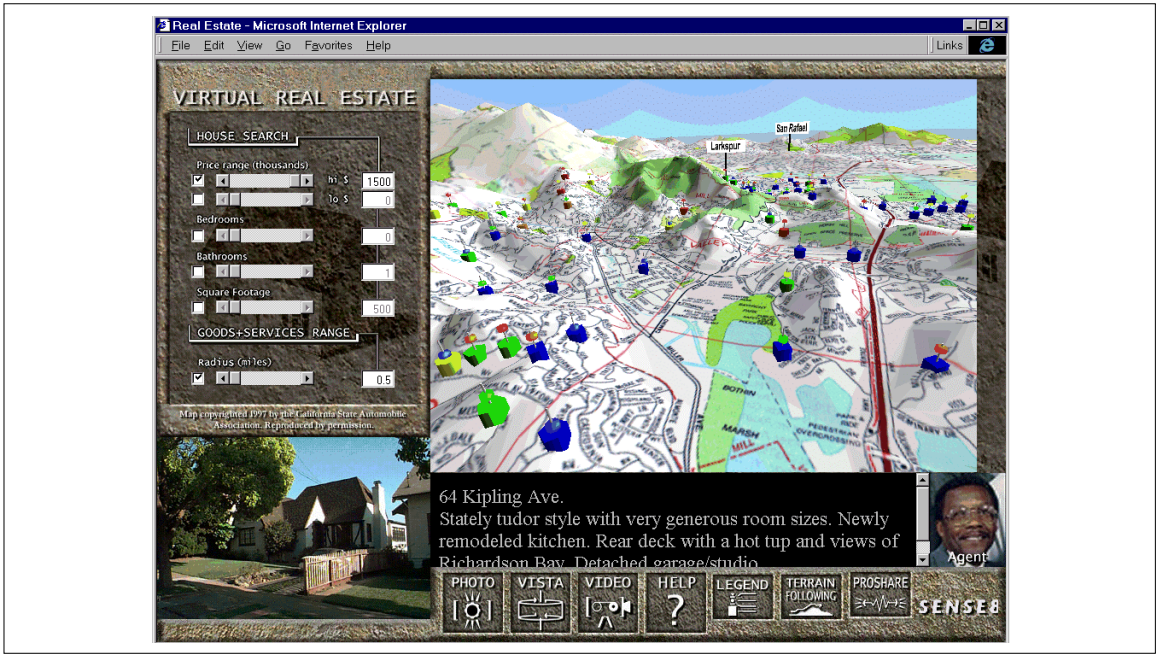
Typically, laptop computers have little hardware acceleration. In this case, it's often better to run your simulation under Direct3D for performance reasons.

Am I using any graphics API specific calls?

If you are using Viewport or RenderNode objects in your simulation, you need to use the OpenGL player since these are not supported under Direct3D.

Embedding Your Simulation

WorldUp provides you with the ability to distribute your simulation as a stand-alone application or embedded as an ActiveX control. The ActiveX control is a powerful mechanism for embedding a 3D interface into a larger application where it is not always appropriate to have 3D control the entire interface, or for where traditional 2D control and/or additional multimedia elements are more appropriate for a greater view.



Real Estate Example

The example in the figure above is of such an application, where browsing the Real Estate market requires both the visual-spatial perception of a 3D world combined with more traditional 2D media components and controls. Using the WorldUp ActiveX control, you can quickly drop your simulation into any container-aware application.

To use the ActiveX Control, you must either install it from the CD, or else from our web site (www.sense8.com). When you install it, WorldUp registers the control with your system. You can now embed this control in any OLE-aware container application, including the Microsoft Office, Internet Explorer, and a multitude of others, including any container application you create using Visual C++ or Visual Basic. For more information on how to embed ActiveX controls within these containers, refer to your container application's documentation.

You don't need to make any specific modifications to a WorldUp simulation to run it in the WorldUp ActiveX control. You do, however, need to make entry points and routines if you wish to

communicate between your control and your container. Like any other ActiveX control, the WorldUp control provides you with a standard set of methods and properties, which you can use to interface with your simulation. The table below lists the properties, methods, and events the WorldUp ActiveX control exposes.

WorldUp ActiveX Control Interface

Properties	Data Type	Description
Filename	BSTR	Specifies current UP file. Changing this value loads a new UP file
WindowObject	BSTR	Specifies which window object in your simulation will be used
Running	BOOL	Specifies whether the system simulation is running or not
ScriptsRunning	BOOL	Specifies whether the scripts in your simulation runs every frame
WantCallBack	BOOL	When set to TRUE, this sends a "Callback" event to the container every frame. It also sends a "LoadingProgress" event to the container while a simulation is loading.
Methods	**Returns**	
RunScript(BSTR Script)	Boolean	Runs a script in the simulation. The script must have a "main" entry point.
RunSubroutine(BSTR Script, BSTR SubRoutine, VARIANT arg1, VARIANT arg2)	Boolean	Calls the SubRoutine within a Script object, passing (optional) two arguments to the subroutine as type Variant.
Step(short number)	Boolean	If simulation is stopped, processes one frame of the simulation.
GetObject(BSTR ObjecName)	WUPOBJECT*	Gets a WorldUp object and returns it inside a wup object wrapper (WUPOBJECT).
Events	**Returns**	
CallBack()	Void	If the WantCallBack property is set to TRUE, this event will be called every frame
ScriptEvent(VARIANT arg1, VARIANT arg2)	Void	This event is triggered by the BasicScript call SendToContainer (See online Help under SendToContainer for more information).
LoadingComplete(BSTR Name)	Void	This event is called when the up file has finished loading. Places the file's name into FileName.
LoadingProgress(float Progress)	Void	If WantCallBack is True, this event will be called several times during file load. The Progress variable will contain a normalized indicator of file load progress.

Note In the table above, BSTR is simply a standard variable type representing a 32-bit character pointer. BOOL is a Boolean variable type. The new type definitions are simply provided for compatibility when communicating across various environments.

The WUPOBJECT type is a special wrapper that contains a WorldUp object. The following table lists the interface to the WUPOBJECT class.

WUPOBJECT Class Interface

Properties	Data Type	Description
SzObjectName	BSTR	Name of the WorldUp object.
Methods	**Returns**	
SetProperty(BSTR PropertyName, VARIANT Value)	Boolean	Set the specified object property to the specified value.
GetProperty(BSTR PropertyName)	VARIANT	Get the property specified
Destroy()	Void	Delete the object
GetType()	BSTR	Get the type of the object (returns a string).
Duplicate	IDispatch*	Duplicate the object
IsDerivedFrom(BSTR szTypeName)	Boolean	True if the object is of the specified type.

Before leaving this section, let's take a quick look at a simple example of how this works. In the following code sample, we use a Visual Basic container from which to examine the position of an object in our simulation at every frame:

```
sub WUP1_Callback( )
  dim myobject as WUPOBJECT
  set myobject = WUP1.GetObject(
  "Avatar" )

  dim position as Single
  position = myobject.GetProperty(
  "Translation")
end sub
```

Refer to a Visual Basic manual for more in-depth coverage of container applications and how to interface with ActiveX controls from Visual Basic.

Distributing Your Simulation over the Internet

To distribute your simulation over the Internet in a browser, you can use either the Internet Plug-in or the ActiveX control. The Internet Plug-in works with both Netscape Navigator and Internet Explorer. However, it offers little flexibility for interfacing between the browser and the simulation. In addition, the user must first install the Internet Plug-in on their machine before they can view WorldUp simulation files.

A much more robust way of handling Internet distribution and viewing is using the signed WorldUp ActiveX Control CAB file. Using ActiveX technology, a new user can encounter WorldUp content on a web site and with one click

download, install, and view your simulation, without requiring the user to leave the web site or reboot their system.

As a developer you can also redistribute our ActiveX Control on your own web page with your WorldUp content. To do this, simply embed the ActiveX control CAB file (located on your WorldUp R5 CD) into your HTML file using the object tag

```
<object
  classid="CLSID:A94D0C23-BE25-11CF-
  A0B7-00A024281615"
  align="center"
  border="0"
  width="256"
  height="256"
  codebase="http://www.mywebsite.com/
  wupcabllocation/
  wupogl.cab#version=5,0,0,0"
  id=wup
</object>
```

If the ActiveX control is not already registered on the user's machine, this will prompt them to download and register the WorldUp ActiveX control. Before distributing the ActiveX control, you should contact us to confirm the version you have is the most current version.

Once the control has been registered on the target machine, Internet Explorer will be able to display WorldUp simulation files. To do this, simply set the Filename property of the WorldUp ActiveX control embedded on the page with the name of a simulation file. For example:

```
wup.Filename = "http://
www.mywebsite.com/wupcontent/
mysimulation.zup"
```

Distribution Checklist

Finally, as a review, before distributing your application confirm the following:

- Does the target machine have a WorldUp player installed on it?
- Do the player and graphics hardware agree (D3D or OGL on Windows platforms)?
- Are you using any OpenGL or Direct3D specific calls in your simulation/plug-ins that require the use of a specific player?
- Are you distributing your simulation as an UP or ZUP file?
- If you are distributing as an UP file, have you set the project paths correctly?
- If you are distributing your simulation as a ZUP file, does the target computer have a previous version of your simulation already expanded on it in its WuCache directory?
- Have you included all DLLs, sounds, scripts, models, and images in your UP or WUP file, including those not instantiated during packaging?
- If you need protection, have you encrypted your BasicScript files?

A

Environment Variables

You may want to configure your computer for maximum performance. Environment variables allow you to optimize the way your operating system interacts with your hardware and the WorldUp software.

To add environment variables, do one of the following, depending on your platform:

- *For Windows NT 4.0* – Choose Settings from the Start menu, then select System and click the Environment tab.
- *For Windows NT 3.51* – Choose Control Panel from the Main Program group, then select System.
- *For Windows 95* – Use the set command in either your autoexec.bat file or another batch file.

Warning Do not change your environment variables unless you are quite certain of the consequences.

WTKZBUFFERSIZE

WorldUp performs its calculations assuming that a Z-buffer of depth 24 exists. Some graphic accelerators only support 16-bit (or less) Z-buffers. If you don't set this value correctly, the hardware graphics accelerator won't work and you'll wind up using the default software OpenGL implementation,

causing a significant drop in performance. You can avoid this by setting WTKZBUFFERSIZE to the depth of the actual hardware Z-buffer. For example:

- Variable WTKZBUFFERSIZE
- Value 16

WTKALPHATEST

Causes pixels, whose final computed transparency value (after factoring in the polygon's material opacity and texture alpha values) is below this value (0-255), to not be written to the framebuffer. This will ensure that all pixels whose transparency value is below a specified threshold value to be treated as completely transparent. This can be useful when you want to have a cookie-cutter effect with your textures. For example:

- Variable WTKALPHATEST
- Value 24

The default is 0 on Windows platforms.

WTKMAXTEXSIZE

Texture images will be shrunk, if necessary, so that the image width and height in pixels will not exceed this value. By setting this environment variable to an appropriate value you can help ensure that your application does not exceed your hardware texture memory limits. For example:

- Variable WTKMAXTEXSIZE
- Value 256

The default is 1024 (this is also the maximum).

WTKSQRTEX

Texture images will be shrunk, if necessary, so that the texture's width and height are equal. The possible values are 0 (zero) and 1 (one), where 0 = off. For example:

- Variable WTKSQRTEX
- Value 1

The default is off (zero).

WTKPROXY

HTTP proxy server (hostname:port). Used when reading VRML files, for example, URLs contained in anchor and/or inline nodes are relative to the proxy server specified here.

- Variable WTKPROXY
- Value BATMOBILE:8080

WTKMULTISAMPLE

Specifies the anti-aliasing sampling rate (must be a power of 2). A higher sampling rate will result in a better quality image but will take more processing time. The default is 0. This variable is used in conjunction with the Anti-aliasing property of the Universe object in your simulation.

Note This applies only to supported hardware. Contact Technical Support for a current list of supported video hardware.

B

WorldUp Players and Plug-Ins

Several players and plug-ins are available for WorldUp that allow you to freely distribute your simulations for non-commercial use. If you wish to distribute your simulations for commercial use, commercial versions of the WorldUp players are also available. These players and plug-ins allow an end-user to view your simulations without having WorldUp installed on their computer. There are two versions of each player, one for OpenGL and one for Direct3D.

This appendix gives a description of the available WorldUp players and plug-ins and describes how to install, distribute, and run them.

Available Players and Plug-Ins

The following WorldUp Players and Plug-Ins are available:

- WorldUp Stand-Alone Player

  The basic stand-alone player that requires no additional software is ideal for re-distributing your WorldUp simulations for non-commercial use. This player (both the OpenGL and Direct3D versions) is automatically installed on your system by the WorldUp installation program.

- WorldUp Internet Plug-In Player

  An internet plug-in player that allows WorldUp simulations to be viewed through browsers like Netscape and Internet Explorer. You can freely distribute the WorldUp Internet plug-in players for non-commercial use. (Currently only available for the Windows platforms.)

- WorldUp Embeddable Player

  An ActiveX plug-in that allows you to embed WorldUp simulations in Visual Basic, Visual C++, Access, and other OCX aware applications. You can freely distribute the Embeddable plug-in player for non-commercial use. (Currently only available for the Windows platforms.)

- WorldUp Stand-Alone Commercial Player

  The basic stand-alone player that requires no additional software is used to re-distribute your WorldUp simulations for commercial use. *(Note that the WorldUp CD does not contain the commercial versions of the players. Please contact Sense8 for information on obtaining the commercial player(s).)*

WorldUp Player Installation

The WorldUp CD contains a directory named Players which contains six self-extracting files. Each self-extracting file represents one of the players described above and can be distributed to end-users. To install one of the players, simply run the corresponding file, and the desired player will be installed on the system. Users who wish to use the Direct3D version of the players must also install DirectX on their system.

Viewing a Simulation Using a WorldUp Player

You can start the player to view simulations by using the program group item that was created when you installed WorldUp (or the WorldUp Player).

On Windows platforms

1 In the WorldUp program group, select WorldUp Player (either OpenGL or Direct3D).

2 From the File Open dialog box, select a simulation file (any file with an .UP, .WUP, or .ZUP extension).

3 The WorldUp simulation is displayed in a window on your screen.

  Note Your universe must contain an application window in order for the simulation to run in a player. For information on application windows, see Chapter 10, *Windows, Viewports, and Viewpoints*.

4 To close the simulation, click the Close button in the upper right corner of the window.

  Note If the application window is set to borderless for this simulation, the control-menu box will not be visible. Choose End Task from the Windows Task List to end the simulation.

You can also start the player from the a file browser in Windows by double-clicking the WUPlayOGL.exe or WUPlayD3D.exe file. Or, by using the command line option with an argument, you can specify the player to be run, as well as the UP file you want loaded into the player.

Important Notes For Direct 3D Users

Direct3D users should be aware that currently the Direct3D version of the WorldUp Players do not support:

- Viewports
- Render Nodes

If you have any of these objects in your simulation, they will not be visible in the Direct 3D players. You should take this into consideration when developing and distributing your simulation.

C

WorldUp User's Group

A WorldToolKit/WorldUp user group has been organized by WorldToolKit/WorldUp customers with assistance from SENSE8. SIG-WTK provides a world-wide electronic forum for the discussion of WorldToolKit, WorldUp and World2World related issus as well as an anonymous ftp site for uploading and downloading WTK/WUP related data.

Participating in SIG-WTK

The following material comes from the original SIG-WTK chairman, Terry Fong. (Note that SIG-WTK was originally formed for WorldToolKit users, but has since expanded to include WorldUp users.)

Greetings, fellow WTK user!

I would like to cordially invite you to participate in SIG-WTK, the WorldToolKit Users' Group. This group provides a contact point for users of EAI/SENSE8 Product Group's WorldToolKit to discuss and exchange information on a variety of topics.

Among these are:

- 3D objects: modeling, importing/exporting to WTK NFF, sharing.

- Sensor drivers: development, reducing lag and latency.
- Managing user interaction.
- Efficient development of virtual environments with WTK.
- Distribution and sharing of virtual environments.
- Improving simulation performance (e.g., frame rate, quality).
- Platform-specific issues (e.g., GL queues on SGI machines).
- Advocating WTK improvements/changes to SENSE8.

Communicating with SIG-WTK

To subscribe or unsubscribe, go to:

```
http://www.sense8.com/support/
forum.html
```

To send a message to all SIG-WTK members, please address it to:

```
sig-wtk@sense8.com
```

WTK/WUP Electronic Archive Policy

The purpose of this site is to facilitate the collection and dissemination of public-domain data for use with WTK/WUP. This archive will not contain any data which is copyrighted, classified, or commercially sensitive to U.S. organizations.

The site is subject to all pertinent laws and regulations of the U.S. government. Regulation of the site will be performed with the following stipulations:

- All connections and file transfers (in and out) will be logged.
- Access to the site will be subject to the approval of the EAI/SENSE8 Group, and restriction may be enforced at any time.
- Abuse of the site, including the import or export of non public-domain data, will result in access denial.

The EAI/SENSE8 Group software developers for WTK/WUP recognize these things about this archive:

- It is likely to encourage and facilitate the use of WTK/WUP by users within and outside of the U.S.
- It provides unrestricted access to public domain data which is usable by WTK/WUP and may offer added value or benefit to WTK/WUP application developers.

Usage

- This site is intended for SIG-WTK related items only. Please do not use this site for other purposes.
- To contribute, please write the files to the ftp/sig-wtk/incoming directory.
- After review, approved submissions will be moved to the appropriate directories.

SIG-WTK: Web Site

The SIG-WTK support web site features:

- An easy way to subscribe/unsubscribe to the forum at:

  ```
  http://www.sense8.com/support/
  forum.html
  ```

- An online searchable support knowledge base for your support questions at: http://www.sense8.com/support/support.html
- Will soon contain an online 3D content and support repository. This will be announced to the SIG-WTK shortly. It's location will also be on our web pages.

D

WorldUp Shortcuts

General

F1	Help
F4	Toggle Rendering
F5	Run
F6	Step Simulation
F7	Stop (only applies when you are running the simulation in the development environment, or you when you are running the simulation as an application but the Development window has the focus.)
Ctrl + N	New Script
Ctrl + O	Open a Script or a Project
Ctrl + Z	Undo last object movement
Delete	Delete the currently selected objects

Development Window

Ctrl + A	Zoom All
Ctrl + M	Zoom To Selected Node
Ctrl + F	Change Modes to Select Mode
Ctrl + ↑	Change active viewpoint direction to Front view
Ctrl + ↓	Change active viewpoint direction to Back view
Ctrl + →	Change active viewpoint direction to Left view
Ctrl + ←	Change active viewpoint direction to Right view

Ctrl + Page Up	Change active viewpoint direction to Top view
Ctrl + Page Down	Change active viewpoint direction to Bottom view

Script Editor

Ctrl-S	Save and Compile Script
Ctrl-F	Find Text
Ctrl-G	Go To Line
Ctrl-C	Copy
Ctrl-X	Cut
Ctrl-V	Paste
F8	Step Over Line of Script (only when debugging)
F9	Step Into Line of Script (only when debugging)

When Clicking in Scene and Type Workviews

Ctrl	Select or deselect targeted item without deselecting others.

Right-Mouse Clicking On Objects and Properties

In the Type Workview and Scene Workview, right-mouse clicking on an object displays a pop-up menu of commands that are applicable to that type of object. In the Property pane, right-mouse clicking on properties displays a pop-up menu for the following commands:

Event Settings	Displays the Event Settings dialog box
Share Property	Displays the Shared Property dialog box (only displays if your universe contains a W2WConnection object)
Copy	Copies the current value of the selected property
Paste	Pastes the copied property value

Cutting and Pasting Properties in the Property Pane

Please note that this does not actually use the clipboard, so your property value will not be available for use elsewhere, nor can you paste in text from a file, for example.

Ctrl + C	With a property selected: Copies the current value of the selected property
Ctrl + V	With a property selected: Pastes the copied property value into the currently selected property. The properties must be of the same type, or the paste will be ignored. You can only paste values for the following types: Integer, Float, Bool, Vect3D, Vect2D and String.

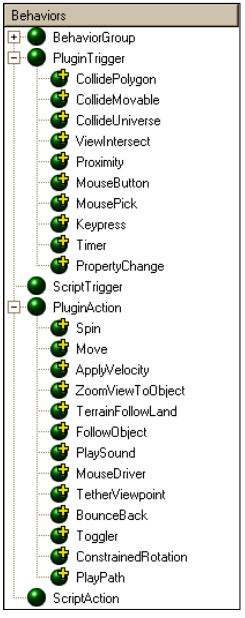
When Dragging Shared Properties in Network Browser

Ctrl	Duplicates the shared property so that is shared under both the sharegroup from which it is dragged and the sharegroup on which it is dropped

E

Pre-Built Behavior Library

WorldUp ships with a library of pre-built behaviors for you to use in your simulations. These pre-built behaviors were built using the WorldUp Plug-in Kit, and are contained in the ActionSet and TriggerSet DLLs in your .\plugins directory. When WorldUp opens, it automatically detects and loads these DLLs, which register the behavior types with WorldUp as shown in the figure to the right.



Several of these behaviors are also available for import as script-based behaviors. These can be subtyped, and their code examined to better understand how a concept is implemented in script, script syntax, or for the purpose of customization. Those that have been ported are available in your .\Behaviors directory. To import them, simply select the desired .PUP file after selecting the Import Behavior  button on the Behavior WorkView.

This appendix gives a brief description of each Trigger and Action in the pre-built Behavior Library, along with a description of each object's inputs and parameters as a reference for both using and creating your own behaviors.

Most behaviors have important properties that you can visually identify with the Important tab on the Property pane. With respect to behaviors, important properties are typically those properties the behavior author felt must be filled in to get the behavior up and running properly.

Plug-in Triggers

COLLIDE POLYGON 1:2

Collide Polygon performs a ray intersect using the input Movable's user-defined cardinal axes as ray direction and the input Movable's global center as ray origin.

Input1	Movable	Source object from which a ray will be cast to determine intersection.
Output1	Movable	Input1.
Output2	Movable	Object containing the polygon the ray intersected.
Search Node	Node	Node to search under in scene graph (default = Root)
Ray Axis	Vect3D	Local axis of Input 1 vector along which to cast ray (default = Z Axis)
Distance Threshold	Float	Distance to poly threshold. Distances below this threshold fire trigger.
Polygon Hit	Integer	ID of polygon intersected by ray.
Distance	Float	Calculated distance to poly intersected.

VIEW INTERSECT 1:2

Collide polygon performs a ray intersect using the input movable's user-defined cardinal axes as ray direction and the input Movable's global center as ray origin.

Input1	Viewpoint	Viewpoint from which ray will be cast
Output1	Viewpoint	Input1
Output2	Movable	Object containing the polygon the ray intersected.
Search Node	Node	Node to search under in scene graph (default = Root).
Distance Threshold	Float	Distance to poly threshold. Distances below this threshold fire trigger
Polygon Hit	Integer	ID of polygon intersected by ray.
Distance		Calculated distance to poly intersected.

COLLIDE MOVABLE 2:2

Collide Movable checks for intersection between two user-specified geometries.

Input1	Movable	Subject Movable
Input2	Movable	Target Movable(s) to check for collision with
Output1	Movable	Input1
Output2	Movable	First Movable in target list (input2) that subject Movable collided with
Ignore Subject Children	Boolean	Check Subject's subtree?
Ignore Target Children	Boolean	Check Target's subtree?

COLLIDE UNIVERSE 1:2

Collide Universe checks for intersection between the object it is attached to and the scene graph. If it collides (or is in collision with) something, it sets the Movable Hit property to the object collided with and fires.

Input1	Movable	Subject Movable
Output1	Movable	Input1
Output2	Movable	Movable in universe subject movable (input1) collided with

MOUSE BUTTON 1:0

Get Mouse event. If event satisfies user-specified event of nine possible button conditions (LEFTDOWN, LEFTHELD, LEFTUP, MIDDLEDOWN, MIDDLEHELD, MIDDLEUP, RIGHTDOWN, RIGHTHELD, RIGHTUP), get the 2D position of the mouse at event time and, if mouse is over geometry, set the triggers' "geometry picked" and "3d point picked". Finally, fire the Trigger.

Input1	Mouse	Mouse Object we'll be getting button events from.
Mouse Event	ENUM	LEFT, MIDDLE, RIGHT + DOWN, HELD, DOWN (i.e. LEFTDOWN)
3D Point Picked	Vect3d	Point (if any) on 3D geometry under mouse' coordinates
2D Coordinates	Vect2d	2d screen coordinates of mouse cursor.

MOUSE PICK 2:1

Fires if the geometry the mouse is over when the button is pressed is in the list of pickables.

Input1	Mouse	Mouse Object we'll be getting events from
Input2	Movable	List of pickable objects
Output1	Movable	Movable picked (if in list of pickables specified in inputlist2)
Mouse Event	ENUM	LEFT, MIDDLE, RIGHT + DOWN, HELD, DOWN (i.e. LEFTDOWN)
Geometry Picked	Geometry	WUP Geometry object picked
3D Point Picked	Vect3d	Point (if any) on 3D geometry under mouse coordinates
2D Coordinates	Vect2d	2D screen coordinates of mouse cursor.

KEYPRESS 1:0

Get the keyboard event and store it in Last Key Pressed.

Input1	Window	Window Object to retrieve keypresses from
Filter Key	String	Keypress condition trigger must match in order to fire. If left empty, any key will fire
Current Key Pressed	String	Current key pressed
Last Key Pressed	String	Last key pressed

TIMER 1:0

Timer fires every *rate* seconds.

Input1	VBase	WUP Object to which the Timer is attached.
Rate	Float	Time (in seconds) between firings.
Reference Time	Float	Internal – used by the timer.

PROXIMITY DETECTOR 2:2

Proximity Detector checks for proximity between a *subject* Movable and a list of *target* Movables. Proximity is measured as the distance between geometry midpoints. If a target is a Group node, it is the computed center of the Group node, including its children. This trigger fires for each object in list of targets the subject is in proximity with.

Input1	Movable	Subject Movable
Input2	Movable	Target Movable to check for proximity with.
Output1	Movable	Input1.
Output2	Movable	Target Movable within proximity.
Distance Threshold	Movable	Distance between midpoints below which trigger fires

PROPERTY CHANGE 1:1

Property Change monitors a particular property common to a list of inputs and fires when that property changes, passing along the object whose property changed.

Input1	Vbase	Object(s) whose property will be monitored
Output1	Vbase	Input1
Property Name	String	Property name to register with the event system.
Always Active	Boolean	If False, fires only when simulation is running

Plug-in Actions

SPIN

Spins an object about a user-defined axis and rate in degrees per frame.

SpinAxis	ENUM	Cardinal axis about which to rotate the object (X, Y, or Z)
ReferenceFrame	ENUM	Coordinate frame in which to rotate the object (local, parent, or global)
PerSecond	Boolean	If true, rotates in Speed/second. Otherwise, it's Speed/frame
Speed	Float	Number of degrees per frame (or second) to rotate

MOVE

Moves an object by the amount specified in the translation property.

Translation	Vect3d	Direction and magnitude to move the object
Reference Frame	ENUM	Frame in which to translate the object (local, parent, or global)

APPLY VELOCITY

Takes the input Movable's Velocity property and applies it to the Movable in a time-based manner.

Reference Frame	ENUM	Frame in which to translate the object (local, parent, or global)

ZOOM VIEW TO OBJECT

Zooms the viewpoint specified by the "Viewpoint" property to the movable in Input1.

Speed	float	rate (in units per frame) at which to zoom to the input1
Distance Threshold	float	Distance from center of input1target at which to stop zooming
Viewpoint	Viewpoint	Viewpoint that will be doing the zooming

TERRAIN FOLLOWING LAND

TerrainFollow causes the input Movable(s) to follow a certain height above a list of user-specified terrain geometries. Orientation of the target is updated to remain in alignment with the ground geometry's surface normal.

DistanceOffGround	float	Distance from center of input to center of ground terrain geometry
GroundObjectRoot	Node	Group node containing one or more terrain geometries

PLAY SOUND

Plays a WorldUp Sound object. This behavior uses the Sound object's properties to determine play characteristics (pitch, repeating, etc.).

Sound	Sound	Any WorldUp sound object currently loaded in the project

FOLLOW OBJECT

FollowObject causes the input Movable(s) to follow the specified Target.

Speed	Float	Rate at which to update follow location
Follow Distance	Float	target distance input (follower) should strive to achieve behind "Target"
Target	Movable	Object to follow

MOUSE DRIVER

Control a Movable's position with the Mouse cursor.

Speed Feedback	Sound	Sound object whose pitch is modulated by the object's speed
Steering Feedback	Movable	Object whose rotation is modulated by the turn angle
Controlling Window	Window	Window that should be checked for mouse messages
Forward Axis	ENUM	Cartesian axis on the input movable this behavior should consider to be "forward"

TETHER VIEWPOINT

Tether Viewpoint tethers (or attaches) a viewpoint to a Movable. Offset allows you to control the position of the viewpoint relative to the Movable's center. If you apply an offset, the viewpoint’s direction will be set to look at the center of the Movable it is tethered to.

Viewpoint	Viewpoint	The viewpoint to tether.
Offset	Vect3d	Vector displacement of viewpoint center relative to movable's center

BOUNCE BACK

Bounce Back reverses a movable along it's z-axis.

TOGGLER

Toggler is a helper behavior that adds it’s input Movable to another behavior’s input list. If the input is already in the other behavior’s input list, Toggler removes it. This has the effect of toggling a behavior on a certain object by adding/removing the object from the input list of a specified behavior.

Target Action	Action	The Action whose input list we wish to add the Active Movable to.

CONSTRAIN ROTATION

Constrain Rotation constrains a Movable's rotation abilities to the user-specified angle sweep in the user-specified coordinate system.

Positive Sweep	Float	Positive angle sweep (in degrees) the object can rotate in
Negative Sweep	Float	Negative angle sweep (in degrees) the object can rotate in
Relative To	Movable	Optional parent object whose coordinate frame should be used to calculate ± sweep (for example, a door's frame).

PLAY PATH

Sets an input Movable position and orientation to a position specified by an element in a Path object. Note this does not change the Path object playing status. It simply uses the Path object's array of element positions.

PathObject	Path	Any currently loaded path object.
MyElement	Integer	Behavior instance specific element number to play.

F

WorldUp File Formats

WorldUp supports the following 3D geometry file formats.

.3DS	Autodesk 3D Studio mesh format
.FLT	MultiGen OpenFlight format
.WRL	Virtual Reality Modeling Language 1.0 and 2.0
.JT	DirectModel (JT) CAD Loader
.NFF	WorldToolKit Neutral File Format
.OBJ	Wavefront OBJ format
.SLP	Pro/Engineer RENDER SLP format
.DXF	Autodesk DXF format

Descriptions for each supported file format are given below.

Autodesk 3D Studio Mesh

WorldUp supports the Autodesk 3DStudio format for Releases 3 and 4. WorldUp reads polygonal information from a 3DS file including color and texture information. WorldUp uses the *ambient* color material value as the color for each polygon, and supports 3DS texture uv values to allow correct reproduction of the 3D Studio texture application

methods. Smoothing groups are supported for Gouraud shading. A 3DS file can contain multiple geometries. The reader does not yet support the following:

- Points, Lines, Splines, Curves
- Face mapping of textures, Box Mapping of textures
- Mirror objects
- Masks

WorldUp does not currently support the 3DStudioMAX file format; however, 3DStudioMAX supplies an exporting tool that allows you to save your files in the *.3ds file format that WorldUp can use.

MultiGen OpenFlight

WorldUp R5 now supports the latest WorldToolKit Multigen FLT reader which supports MultiGen files greater than V14.2 through V15.5. This is shipped as a separate product. The FLT reader supports textures and transforms with other records as shown below.

Supported records

- Material Palette
- Texture Palette
- Object
- Group
- Group with animation 1
- Light source records (Infinite, Point, Spot)
- Level of Detail
- Subfaces
- Switch 2
- External Reference
- Instance

Note A MultiGen animation record is translated to a WorldUp switch node. Each frame of the animation sequence is a child object of the switch. The first frame is the default active child. Translated MultiGen switch nodes do not maintain a list of masks. The default active node under the resulting WorldUp switch node will be the first child of the switch node.

Unsupported records

- Header
- Eye point
- Light point
- Binary Space Partition
- Curve
- DOF
- Sound
- Text
- Road
- Path

Note

1. Any WorldUp simulation using the FLT reader must also distribute all of the MultiGen API DLLs.
2. Primary colors are applied to polygons only if there is no material applied to the polygon. Secondary colors are unsupported.
3. Material properties are always blended with textures.
4. If a texture specified in the MultiGen file is missing, a texture representing a red X on a white field will be applied in its place. The user can change this texture by replacing the existing NOTEX.TGA image located in the WorldUp/images directory with one of their own creation.

5 A separate material table is created for the MultiGen file and each external reference.

6 The name of the table is the name of the file with an MT appended to the front and missing the .flt suffix. For example, the externally referenced file TEST.FLT will have a corresponding material table called MTtest.

7 Material table indices in WorldUp will be one greater than the same entry in the MultiGen material palette. This is to allow the addition of a default material at index 0 for those polygons without a material or color.

8 If your geometry is not visible in a shaded rendering mode, you may have a 100% transparent material applied to it. Check the material table entries in your modeler.

Virtual Reality Modeling Language (VRML)

WorldUp can read and write VRML 1.0 and VRML 2.0 (.wrl) files.

VRML 1.0

WorldUp supports most of the VRML 1.0 specification. The VRML 1.0 limitations of WorldUp include:

- No support for AsciiText, FontStyle, IndexedLineSet, and PointSet nodes.
- The crease angle field within ShapeHints nodes is ignored.
- WorldUp ignores scaling factors (if any) within a Transform node's transformation.
- WorldUp can read and process geometric primitives (such as cone, cube, cylinder, and sphere), but they are internally decomposed into polygons (i.e., they are not internally retained as cone, cube, cylinder and sphere primitives).
- WorldUp uses its own convention to apply textures to faces without texture coordinates
- WorldUp support for instancing (USE/DEF scheme) does not include all node types. The Coordinate3, Material, and Normal node types cannot be instanced unless they are in the same scope (for example, there is no separator that differentiates the state of one instance from that of the other).

VRML 2.0/97

WordUp supports basic geometry within the VRML 2.0/97 specifiication. It is not intended to support any behavior or other advanced node types (such as sound). The following gives a list of the limitations for VRML 2.0/97.

- Limitations of Supported Common Node Types:

  ImageTexture – Remaining issue between texture blend and texture decal. Use Texture blend.

  IndexedFaceSet – If a colorIndex is specified this will override the use of textures. Polygon are limited to 256 vertices. In addition each vertex may be shared by a maximum of 64 polygons.

  Lights (DirectionalLight, PointLight, SpotLight) – Point lights and spot lights work best, however directional lights are supported, though the desired lighting effect maybe slightly off.

  Material – Do not specifiy (export) a material color if you want the geometry to be textured. Textured and colored geometry is not currently supported.

PROTO – Be careful using this type, it is not currently fully supported.

TextureCoordinate – Do not specifiy a material color if you want texture coordinates to work correctly.

USE/DEF – Do not create recursive DEFs. That is:

```
DEF Mytype {
  USE Mytype
    …
}
```

Viewpoint – This works, but currently sets the active viewpoint each time this node type is encountered. Hence a file with 10 viewpoints will cause the importer to change the active viewpoint ten times

- Unsupported Node Types:

  If any of these appear in the file most will be discarded as expected.

  Audio Clip - Discarded

  Background - Discarded

  Billboard - Treated as group node.

  Collision - Treated as group node.

  ExternPrototype - Discarded

  FontStyle - Discarded

  IndexedLineSet - Discarded

  Interpolator - Discarded

  NavigationInfo - Discarded

  PointSet - Causes WTK to crash in one case. DO NOT USE!

  Routes - Discarded

  Script - Discarded

  Sensors (CylinderSensor, PlaneSensor, ProximitySensor, SphereSensor, TimeSensor, TouchSensor, VisibilitySensor) - Discarded

  Sound - Discarded

  Text - Discarded

  WorldInfo - Discarded

Exporting a File as VRML1.0

If you are planning to export your scene graph in the VRML format, you will need to ensure that all of your textures are stored as JPEG files. This is because web browsers do not support *.rgb or *.tga files. They require JPEG or GIF IMAGE files (GIF images are currently unsupported by WorldUp).

CAD Loader (DirectModel or JT)

WorldUp now has CAD loader solution which can load DirectModel (.jt) files. This is shipped as a separate product.

Notes on CAD Loader

- Geometry Reconstruction - At the geometry level in the JT file format, vertex information is organized in triangle strips for rendering performance. Thus, the associated geometry cannot be efficiently edited at the vertex level (for example, vertices are not shared across strips so you cannot move one vertex without creating a hole in the model's mesh). WorldToolKit's jt2wt file loader actually processes these vertices and reconstructs polygons out of them. Duplicated vertices are collapsed into one shared vertex, so the resulting database is efficient.
- No support for textures.

WorldToolKit Neutral File Format (NFF) and Binary NFF

The NFF format is an efficient and readable representation of 3D geometry. It is also useful as an intermediary format between WorldUp and formats not otherwise supported. An NFF or binary NFF file can contain multiple geometries.

Wavefront OBJ

The Wavefront modeling tool generates this format. WorldUp imports the 3D polygonal geometry and curved surfaces that have been polygonalized. Vertex normals and texture vertices are supported for Gouraud shading and texture draping. WorldUp reads map files and material files, but the only supported properties are diffuse color (Kd) and diffuse texture (map_Kd). An OBJ file describes a single geometry.

Pro/Engineer RENDER SLP

WorldUp reads the facets in an SLP file as colored polygons with vertex normals for smooth shading. A SLP file contains only one geometry.

Autodesk DXF

Many CAD packages such as AutoCAD and other 3D modeling programs generate this common format.

You can load in many other geometry files into WorldUp using third-party geometry conversion programs capable of writing formats that WorldUp can read. A program, such as PolyTrans, reads and writes most popular 3D file formats.

G

Glossary

3D Sound	Spatialized sound that appears to the end-user to have a distinct location in the simulation.
3DS	The native file format of Autodesk's 3D Studio. You can use this binary file format to represent 3D geometry, lighting, and animation.
6D Sensor	Sensors that have six degrees of freedom of movement. That is, they can control movement in the X, Y, and Z direction, i.e. can control pitch, yaw, and roll.
Absolute Record	Sensor values that correspond to a specific absolute spatial location (i.e. the position and orientation of the sensor). See also Relative Record.
Active Moveable	The movable currently occupying a behavior "input X" slot.
Active X Control	A powerful mechanism for embedding a 3D interface into a larger application where it is not always appropriate to have 3D control the entire interface, or for where traditional 2D control and/or additional multimedia elements are more appropriate for a greater view. With this, you can embed your simulation inside of ActiveX Control Containers, such as Visual Basic, Internet Explorer, or Macromedia Director. Exposes an interface that allows communication between control and container.
Ambient Color	The material property that represents the color reflected from a material in shadow.
Ambient Light	Background light that illuminates all graphical objects equally, regardless of their position or orientation. By default, ambient light is always present in a WorldUp application.
Ancestor Node	Any node whose sub-tree contains a node (N), is considered to be an ancestor of that node (N).
Anti-Aliasing	The process of reducing aliasing, or jaggies, in creating an image.
Application windows	The windows in which the simulation displays when you run the simulation as an application within WorldUp, or when your end-users run the simulation using one of the WorldUp players. Application windows are created from the Window object type.
Attenuation	The degree to which a point or spot light's intensity decreases with increasing distance from the position of the light.

Axis	An axis represents a reference line of a coordinate system along which a geometry is translated, or around which a geometry is scaled or rotated. There are three axes (X, Y, and Z) representing width, height, and depth.
Back Face	The back face of a polygon is the side facing away from the direction of the polygon normal (or if the polygon does not have a polygon normal, the back face is the side from which the polygon's vertices appear in clockwise order).
Back Face Rejection	The elimination of a single-sided polygon (that is, a polygon that can only be viewed from one side) from the rendering process. In the back face rejection process, those polygons whose normals face away from the viewpoint (or whose vertices appear in clockwise order) are not rendered.
Base Window	All viewports must have a base window which acts as frame to hold a set of viewports. A Base window can contain up to 8 viewports.
BasicScript	A scripting language which is syntactically identical to Microsoft's Visual Basic.
BasicScript Encryption	A binary version of a script file which can be generated by WorldUp so that the contents of your scripts can remain hidden even when you distribute your simulation by deploying your .WUP or .UP file. Encrypted BasicScript files have an .EBC extension, while BasicScript ascii files have an .EBS extension.
Baud Rate	Data transmission speed in bits per second.
Behavior State	A Behavior has 3 defined states: • READY (0) Behavior is ready to fire (could be disabled though through its Enabled property) • FIRING (1) A Behavior is currently in it's or one of it's children's callbacks. • SPENT (2) Behavior has fired, and as a result of its Repeating property set to false, is not ready to fire again. This typically occurs in "one shot" behaviors.
Behavior System	• Provides a mechanism for visually assembling simulations rather than programming them. • Provides a natural learning pathway for increasing simulation complexity • Provides a redistribution mechanism so that users can both provide and benefit from other pre-built behaviors.
Behavior Wizard	The interface that enables you to author your own script based Triggers and Actions. A series of dialogs the steps you through the process of creating a new Behavior type.
Behavior Workview	The Workview where all Behavior creation and scheduling is done. This Workview is comprised of four panes which together allow you to create a new behavior, schedule the behavior, add inputs to the behavior, edit the behavior's properties, and finally export the behavior for re-use.
BFF	The binary version of SENSE8's neutral file format used for representing 3D geometry. See NFF.

Bothsides (of a polygon)	Polygons have front and back sides (or faces). The side facing in the direction of the polygon normal (or if there is no polygon normal, the side from which the vertices appear in counter-clockwise order) is considered to be the front facing side. You can choose to display just the front side of a polygon or bothsides of a polygon. If a polygon is bothsided, it can be viewed from either side.
Bounding Box	Also known as Extents Box (smallest box that surrounds an object). The term bounding box sometimes refers to the fact that extents boxes can be made visible in the scene.
Breakpoint	A user-specified line in a script at which WorldUp will stop running the script. You set breakpoints to help debug your script. For example, if you set breakpoints on lines 4 and 7 of the script and the script runs correctly up to the first break point, but fails before the second breakpoint, the failure is somewhere between lines 5 and 7.
Callback	A Behavior's actual subroutine (C or BasicScript) that represents the Behaviors actual function.
Casting	Mechanism in BasicScript by which objects can be assigned to variables whose type is not identical to the object's type. To use casting, the object's type must be a subtype or supertype of the variable's type.
Centroid	The centermost position of a three-dimensional object.
Child Node	A scene graph node that is a direct descendent of another (parent) node. A child node can inherit state information from its ancestor nodes.
Collision Detection	Intersection testing of objects at either the bounding box level or at the polygon level.
Concave Polygons	Any polygon that has at least one interior angle greater than 180 degrees.
Coordinate System	A positional system, containing X, Y, and Z components, by which three-dimensional entities can be described. See Local Coordinate System, Parent Coordinate System, and World Coordinate System.
Coplanar Polygon	Polygon surfaces that overlap and lie in the same plane.
Culling	See Hierarchical Culling.
Cylindrical Mapping	A technique for applying texture mapping coordinates so that the image appears to be wrapped around the object in a tube-like fashion. The application of a label to a can or bottle is an example of cylindrical mapping.
Data Point Entry Mode	The mouse, when used with the WorldUp Modeler, is in one of two modes at any one time: viewpoint manipulation mode or data point entry. Data point entry mode allows you to create, edit, and select the components of your model.
Descendant Node	Any node that is contained in the sub-tree of another node is considered to be a descendant of that node.

Development windows	The application window in the application editor, where you can manipulate your Node objects as you develop the simulation. Development windows are created from the DevWindow subtype, located under the Window type.
Diffuse Color	The material property that represents the color reflected from a material in direct light.
Diffuse Light	Positional or directional light that illuminates polygons as a function of the angle between the light direction and the polygon (or vertex) normal.
Direct Model (.jt files)	A cross-platform large CAD model rendering toolkit
Directed Light	A light source that has direction but no (finite) position. A directed light can be used to emulate the effects of sunlight.
Distributed Simulation	A simulation that is shared between multiple users across the network.
DLL	A Dynamic Link Library is a software library which dynamically links to an application at runtime.
DOF	Degrees of freedom. See 6D Sensor.
Downstream	A descendant to a Behavior's is said to be downstream.
DXF	Drawing Interchange Format. This file format was developed by Autodesk, Inc. as a way to transfer geometric data from one design application to another.
Emissive	A material property that represents the color produced (not reflected) by the material even when there is no light. A geometry with this property can be seen even when there are no lights in the scene, however, the emissive light does not illuminate other geometry in the area. This material property is used less often than the others
Encrypted BasicScript Files	See BasicScript Encryption.
Euler	A mathematical representation of a position and orientation in three-dimensional space.
Extents Box	The extents box is the smallest box that fits around an object. See also Midpoint and Radius.
Extrusion	The 3D outline or object created by taking a 2D contour and extending it into three dimensions.
Facet Mapping	A technique for applying texture mapping coordinates so that the image appears on each polygon of a object. For example, if facet mapping is applied to a cube, each facet of the cube would look the same.

Flip Normals	A polygon has a front side and a back side (face). The side of the polygon facing in the direction of the polygon normal is the front face. The side of the polygon facing away from the direction of the polygon normal is the back face. (See Backface for information regarding polygons which do not have polygon normals.) The Flip Normals command in the Modeler swaps a polygon's front and back faces.
FLT	MultiGen/ModelGen Flight file format used to represent geometric objects. FLT files can contain multiple geometric objects and may contain textures and LODs.
Fog Node	A scene graph node used to simulate fog, smoke, etc.
Frame	An individual rendering loop during which each active window is redrawn after updating sensor input, path information, and running scripts associated with simulation objects. See also Simulation Loop.
Frame of Reference	Allows you to select a reference frame (coordinate system) about which objects are translated and rotated. Four reference frames are available. World, Parent, Local and View. World: Object manipulation is performed in the universe's reference frame. Parent: Manipulation is performed relative to the frame of the node's parents. Local: Manipulation is performed relative to the selected node's local frame. View: Manipulation is performed relative to the viewpoint's frame.
Frame Rate	The number of times per second that WorldUp completes the simulation loop, i.e. renders a frame.
Front Face	The side of the polygon in the direction of the polygon normal. The front face of a polygon which does not have a polygon normal, is the side from which the polygon's vertices appear in counter-clockwise order.
Geometry Node	A scene graph node, such as block, sphere, cylinder, text3d, or imported, which is used to model physical objects contained in the scene.
Gouraud Shading	A technique used for shading a 3D graphical object composed of polygons, by interpolating light intensities at the vertices of each polygon's face, rendering a smooth surface.
Graphical objects	Objects created from the Geometry object type or one of its subtypes (Block, Cylinder, Imported, Sphere, and Text3d).
Graphics Pipeline	Many high performance systems utilize specialized graphics hardware (aka graphics pipeline) to substantially increase the system's ability to process and render geometric objects composed of polygons.
Group Node	A scene graph node that has children and is used to organize components of the scene graph in a logical manner.
GUI	Acronym for graphical user interface. Also called user interface (UI).
Head Mounted Display	A display device that is worn on the head, which sometimes permits position and orientation tracking.

Heads Up Display	The static portion of an image rendered on a display device.
Hierarchical Culling	WorldUp's automatic process of quickly and efficiently eliminating objects that are not visible from the viewpoint so that they are not unnecessarily processed during the rendering process.
Hierarchy	Used in the context of scene graphs, hierarchy refers to how the nodes in a scene graph are organized and the relationship of one node to another. Used in the context of the Type Workview, hierarchy refers to how object types are sub-classed from other types.
Hither Clipping Plane	The physical range in front of the viewpoint, before which objects are not rendered in that window. That is, objects that appear between the viewpoint and the hither clipping plane are not rendered. Objects are rendered only in the area between the hither clipping plane and the yon clipping plane. Hither Clipping is a Window property.
HLS	HLS stands for hue/luminance/saturation.
Input Slot	The slot that hold a Behavior's "Active" input, which is the object the Behavior is currently acting upon.
InputList	A list of WUPObjects set by the behavior user to define the list of objects that a behavior will act upon. The Task Scheduler uses this list to fill slot "in1" as such, "InputListX" and "inx" will always have the same type. This list is accessed either through the IDE, or with the behavior functions AddToInputList(int, Movable) and RemoveFromInputList(int, Movable)
Instance	WorldUp's scene graph hierarchy allows geometry nodes to be referenced multiple times within a scene graph. Since it is sometimes necessary to identify a particular occurrence of a node to distinguish it from other occurrences, each occurrence is called an instance.
Interpolation	The method of determining a new value using two or more existing values. WorldUp uses interpolation when new paths are created from previously defined paths.
Intersection Testing	See Collision Detection.
Iterators	Iterators are script variables used to cycle through objects of a given type. Search on "GetFirst" and "GetNext" in the online help.
Leaf Node	A scene graph node that has no descendants.
Level of Detail (LOD) Node	A scene graph node used to automatically select between different representations (levels of detail) of an object based upon the distance between the object and the viewpoint position.
Light Node	A scene graph node used to specify a light (point, directed, or spot).
Material	A material is used to define the appearance of graphical objects and consists of the following material properties: ambient color, diffuse color, specular color, shininess, emissive, and translucency.

Material Table	Used to store the material properties of any number of materials. Each geometric object references a number of materials from the material table that is associated with that object.
MaterialNode	Is the most flexible way to add material to an object. The MaterialNode allows materials to be created, edited and saved out for reusing in the Development Environment.
Matrix	A 3x3 or 4x4 array of floating point numbers which is a mathematical entity that can be used to represent position and orientation in 2D/3D space.
Mesh	A group of polygons that share vertices and define a complex surface, such as a curved hood on a car. In the Modeler, meshes can be selected as a unit by using Select by Connected.
Midpoint	The center of a node's extents box. See also Extents Box and Radius.
Model Workview	The import central for importing models into WorldUp simulation.
Motion Link	Used to connect a source of position and orientation information (a path or sensor) with a target that moves to correspond with that changing set of information. A target can be any Viewpoint or Movable object.
Movable Node	A scene graph node that represents self-contained entities like geometries or lights that can be easily moved around in the scene.
NFF	Neutral File Format, SENSE8's neutral ASCII file format used for representing 3D geometry.
Node	The fundamental element or building block used to construct a scene graph. A node is simply an element of content (like a geometry or light), state (fog), or a grouping/ procedural element (group, switcher or lod) used to maintain scene hierarchy.
Node-locked License	A WorldUp software license designed to be used on a stand-alone computer.
Normal	A direction vector used for shading and rendering. Normals can be applied at both the vertex and polygon level. A polygon normal is perpendicular to the polygon surface and extends outward from the visible side of the polygon. A vertex normal represents the direction that is perpendicular to the tangent vector at the vertex position of the polygon.
Normalized	A normalized vector is a vector whose magnitude is 1.0.
OBJ	Wavefront/Alias file format used for representing 3D geometry.
Objects	Are one of the core building blocks upon which your simulation is built. Graphical objects are the objects that you can see in the Application window. The objects in a scene/simulation are what ultimately gets drawn.
Object Types	The classes of objects that can be instantiated or sub-typed. Each object type has a number of properties which describe the characteristics of the object type.

Opacity	See Translucency.
Optimization	A technique used to optimally organize the contents of an imported geometry node so that it can be rendered in the shortest amount of time. Once an imported geometry node's optimized flag is set, you will not be able to make edits to the imported geometry, unless you unset the optimized flag.
Orphaned Nodes	Nodes that are not contained in the Scene Workview but are in the Type Workview are considered to be orphaned nodes. This can occur if nodes are removed from the Scene Workview.
Orthogonal Viewing	A 2D view of the universe (helpful for fine tuning object positioning.)
Orthographic Projection	Orthographic projection is a window property that can be set if you want plan views or anytime a perspective distortion is not desired; parallel lines remain parallel regardless of viewpoint position. Translations in the X and Y directions work as before, but translations along the Z-axis do not affect the scene. If this window property is not set, the view seen will be a 3D perspective projection. See Perspective Projection.
Output Slot	The slot that holds a Behavior's "Active " output, which is the object the Behavior has most recently written.
Parallax	A property of the Viewpoint object type which represents the distance between the left eye and the right eye position when using a stereo viewing device.
Parent coordinate system	The coordinate system of the parent object in the scene graph.
Parent Node	A node's direct ancestor in the scene graph.
Path	Stores a series of position and orientation records in absolute world coordinates. A path can be used to pre-program a flight path through a scene, or to pre-define the motion of an object within the scene.
Path Element	A single position and orientation record. A sequence of path elements defines a path.
Perspective Projection	Perspective projection is a window projection type that is used to display objects in three dimensions (height, width, and depth). By default, a window's orthographic property is FALSE, and hence a perspective projection will be used in each simulation. See Orthographic Projection.
Perspective Viewing	A 3D view of the universe.
Picking	The ability to select the front-most rendered polygon (i.e. object) in a window.
Pitch	The orientation of an object about the X axis.
Pivot Point	The point around which vertices rotate and are scaled. The default pivot point is at the center of the object or set of selected vertices.

Pixel	A contraction of “picture element,” it refers to one point in a graphics image on a computer display. A standard VGA display might have 640 x 480 pixels. The number of bits per pixel determines how many colors can be represented on the image. VGA displays typically have eight bits per pixel. “Truecolor” displays typically use 24 bits per pixel.
Planar Mapping	A technique for applying texture mapping coordinates so that the image appears to be projected through an object.
Planar Polygon	Polygons whose vertices are all positioned within the allowable distance (0.004)) from the plane passing through the vertices.
Plug-In SDK	is a set of high level C functions simulation authors can use to extend the functionality of WorldUp.
Plug-Ins	Are created with R5’s optional Pug-in Kit. Plug-ins are custom simulation objects that directly interface with the WorldUp Object System using a high level set of object management routines. Plug-ins can be made re-usable.
Point Light	An omni-directional source of lighting capable of being positioned by the user. A light bulb is an example of a point light source.
Polygon	A polygon is a planar surface defined by a set of three or more vertices. It is the basic building block of geometries. Polygon properties include material, texture, bothsides, and smooth.
Polygon Normal	See Normal.
Port	A logical channel in a communications system. See also Serial Port.
Portability	The ability to move a simulation built using WorldUp from one hardware platform to another platform without having to recompile or make extensive changes to the simulation.
Position	The current X, Y, and Z coordinates of an object.
Pre-Built Behaviors	A basic library of Triggers and Actions most commonly found in interactive simulations.
Predecessor Node	Any node in a scene graph that can directly affect how a specific node (N) is processed is considered to be a predecessor of that node (N), even though that node is not an ancestor node.

Primary Input	A WUPObject that is acknowledged as a primary, required, input to a behavior's callback function. The Behavior author defines 0, 1 or 2 inputs. The 1st input ("in1") is assumed by the Task Scheduler to be the current WUPObject that the Behavior will act upon. The Task Scheduler fills in this property based on either the Behavior's parent's out1 property or from the user-defined list of WUPObjects ("InputListX"). Primary inputs implemented as a special property set that includes the properties "inX", "InputListX", and "UsesParentX." "InputListX" and "InX" are of the same type, and are defined by the Behavior Author. These properties are not manipulated directly, but indirectly via the GUI.
Primary Output	A WUPObject that is acknowledged as a primary, required, output parameter of a behavior's callback function. The Behavior Author defines 0, 1 or 2 outputs. Primary Outputs are defined by the Behavior Author and are added during Subtyping. Primary Outputs are implemented as "out1" and "out2". They are read-only.
Primitive (Geometric)	A three-dimensional basic geometric form (such as a block, sphere, or cylinder) stored as a collection of polygons.
Project Workview	The overall tabbed interface that brings often used features of worldup into the user's view. The project Workview has a certain row of tabs along the top of it that select a certain 'view of the project'. These tabbed views are the scene, model, behavior and type Workviews.
Projection Modes	Defines how the scene is projected onto the display device. See Orthographic Projection and Perspective Projection.
Propagation of state	When processing the scene graph tree, the lighting and positional state created by each geometric or light movable accumulates (propagates) as the remainder of the scene graph is processed.
Property	Object types and instances of objects have a set of properties which defines the object's characteristics. The actual property values assigned to an object's properties is what distinguishes two objects of the same type. Each property is typed, that is it can be an integer, float, or any other WorldUp supported type.
Property Pane	The pane helpful in viewing and changing the properties of an object. The property pane has four tabs; All, Important, Subtype and Editable. "All" tab shows all the properties of the selected object. "Important" tab just shows the most specific and basic properties of an object which are required to create the object. "Subtype" shows all the properties that exist in the derived type that doesn't exist in its parent. "Editable", as the name says, lists all the properties, which can be edited. In "All" tab the properties shown in Red color are protected or read-only which can not be changed.
Quads	Four-sided polygons.
Quaternion	A mathematical representation of an orientation.
Radius	The distance from the midpoint of a node's extents box to a corner of the box. This is the same as the length of the extents vector. See also Extents Box and Midpoint.

Ray Casting	A ray is a vector representing a direction. Ray casting is the process of calculating a ray that emanates from a position (for example, a viewpoint) and which then passes through a specified point. Ray casting can be used for terrain following or intersection testing.
Real-time Simulation	A 3D application (such as a WorldUp simulation) that responds to input and displays the corresponding change (almost) instantly. When measured in frames per second, real-time usually means at least 10 fps.
Reflection Mapping	Cues to the viewer that determine the spatial relationships between objects. The way visible surfaces are reflected, takes into account light sources, surface characteristics, and the positions and orientations of the surfaces and sources.
Relative Record	Sensor values that correspond to a sensor's change in spatial location (position and orientation) since the last time through the simulation loop. See also Absolute Record.
Rendering	Generation of a graphical image from mathematical models of three-dimensional objects, i.e. a scene.
Rendering Settings	The Project's Rendering Settings for displaying geometries in a simulation. Types of rendering include wireframe, shaded, textured, and textured with perspective correction.
RenderNode	Allows you to create your own custom RenderNode type with the optional Plug-in Kit from Sense8. The RenderNode is a node in the Scene Graph that calls back to a user defined function in every frame during traversal of the Scene Graph. The user-defined function contains low-level drawing commands the user can execute, allowing the user a greater flexibility than is offered by the other physical objects in the Scene Graph. The benefit of RenderNode existing in the Scene Graph is that the node can accumulate the state of the Scene Graph, including lighting and transformations.
Resources	A resource is a file that contains objects, types, or geometry entries that can be extracted and used in the current simulation. Any geometry file of one of the supported file formats, or any universe (.UP) file can be opened as a resource.
RGB	RGB stands for the red, green, and blue components of a color specification. Valid values for color components range from 0 to 255. An RGB triple of (255, 0, 0) represents the color red while an RGB triple of (255, 255, 0) is yellow.
Right-hand Rule	The WorldUp coordinate system obeys the right-hand rule. The default coordinate system has the X axis pointing to the right, the Y axis pointing down, and the Z axis pointing straight ahead.
Roll	The orientation of an object about the Z axis.
Root Node	A scene graph node that is the top most node in any scene graph. A scene graph has only one root node. All other nodes in the scene graph are descendants of the root node.

Rotation	The turning of an object so that it has a different orientation.
Scene	The virtual world being displayed.
Scene Graph	The spatial organization and relationship of Node objects to each other is controlled by your scene graph. A scene graph is a hierarchical arrangement of nodes, organized beneath a single Root node. In WorldUp, you view and modify your scene graph with the Scene Workview. The order in which nodes appear in the Scene Graph pane determines the order in which nodes are processed and the order in which graphical nodes are rendered.
Scene Workview	The most commonly used Workview for creating new objects, editing their properties, scene assembly and scene graph layout. The Scene Workview consists of Nodes pane on the left, Scene Graph pane on the right and Property pane at the bottom.
Script	A script is a collection of one or more functions written in BasicScript that exist in a script file to add behavior to your objects (such as animating objects, detecting collisions between objects, etc.). Script files have an .EBS extension. WorldUp uses two kinds of scripts: Stand-Alone and Task.
Script-Based Behavior	A BasicScript Behavior. All BasicScript Behavior's are comprised of 3 parts: 1. A Script Behavior Object 2. A Script Handler Object 3. A Script File
Script Handler	The WUP Script Object associated with a WUP BasicScript Behavior
Selection Sets	Selection sets are temporary (per modeling session) collections of vertices or polygons of a geometry assigned to one of three letter buttons (A, B, or C) in the Select Menu (or toolbar) of the Modeler. Selection sets allow you to manipulate and assign commands to groups of elements within a geometry. For example, by making all of an object's polygons part of a selection set, you can copy them all at once.
Sensor	A device that responds to physical movement and that transmits the resulting position and (possibly) orientation information.
Sensor Sensitivity	The scale factor associated with a sensor's translational record, i.e. the maximum magnitude of translational input along any axis in any pass through the simulation loop.
Serial Port	A connector on a computer where you can attach a serial line connected to peripherals that communicate using a serial protocol.
Shading	The process of rendering polygons, especially when using lighting effects. See Gouraud Shading.
Shaded Texture	When you enable the texture shading feature, texture colors are affected by lights in the scene. If colored lights are used, the color of texture elements is also affected.

Shininess	A material property that controls the narrowness of focus of specular highlights. This has no meaning if the specular color is black (lighting of geometry rendered with material properties is an “additive” process; a black specular highlight will not darken the geometry; it simply won't contribute to a light highlight on the geometry). The lower the shininess value, the more “spread out” the highlight; the higher the shininess value, the sharper the highlight. A high value for shininess makes an object look shiny.
Shutdown Script	The script that is run each time you close a universe (.UP).). The Shutdown script for a universe is indicated by the Shutdown Script property.
Sibling Node	Children of the same parent node are siblings.
Simulations	The 3D/VR applications that you can build using WorldUp.
Simulation Loop	When a WorldUp simulation is running, the simulation loop is repeatedly executed. WorldUp reads input sensors, updates objects with sensor input, executes object tasks and scripts, steps any paths, and renders a new view of your scene into the simulation window(s) during each pass through the simulation loop. Each pass through the simulation loop is called a frame.
Six Degrees of Freedom	See 6D Sensor.
SLP	Pro/Engineer file format used for representing 3D geometry.
Spanning	If true, it indicates a Behavior/Action spans multiple frames versus steady state. Zoom to is an example of a behavior that spans.
Spatialized Sound	See 3D Sound.
Specular Color	A material property that represents the color reflected from the highlights of the geometry. The specular material property is what makes a geometry appear to be “shiny” with highlights appearing on its surface. Usually, the specular highlight is white, which means that it reflects the color of the specular light (which is also usually white).
Spherical Mapping	A technique for applying texture mapping coordinates so that the image appears to be wrapped around the object in a spherical fashion. A good example of spherical mapping would be a world globe.
Spot Light	A light source that illuminates a small area, within a cone of a specified angle. An automobile headlight is an example of a spot light source.

Stand-Alone Script	A Stand-Alone script contains a Main subroutine. Scripts can contain any number of routines, but only scripts that contain a Main subroutine are Stand-Alone scripts. A script can only have one Main subroutine (In fact, a script cannot contain more than one routine with the same name). You can run any Stand-Alone script independent of the simulation (that is, the script does not have to be attached to an object). Additionally, in WorldUp, you can designate particular Stand-Alone scripts to be your Startup, Shutdown, and User scripts. Stand-Alone scripts are also used to define the action for Navigation Bar buttons.
Startup Script	The script that is run each time you load a project (.UP). The Startup script for a universe is indicated by the Startup Script property. Startup scripts are useful for loading .DLL files.
State	Refers to the accumulated lighting and positional state that results during the processing of a scene graph. The scene graph state affects how and where geometry is rendered at any particular point in the scene graph.
Stereoscopic Viewing	The visual effect achieved when part of your scene appears to be in front of your display screen, and part of the scene appears to be behind your display screen, giving the illusion that the image is a 3 dimensional image.
Subfaces	ModelGen and MultiGen permit “subfaces,” polygons that generally are oriented in the same plane as another polygon, but that are intended to appear as if they are on top of the other polygon. When polygons with subfaces are translated literally into the WorldUp viewing format, Z-buffer roundoff becomes pronounced, resulting in flickering between the coplanar faces as the object is rendered. When WorldUp encounters subfaces in an OpenFlight file, it translates them by a constant amount in the direction of the parent polygon’s normal vector.
Sub-tree	A node and all its descendants in the scene graph is called a sub-tree of the overall scene graph tree.
Sub-type	Lists only the properties of the subtype which are not present in the parent type. When you create a subtype, it inherits all of the properties of the type from which it was derived and becomes subordinate to that type.
Switcher Node	A scene graph node that allows the user to control which of its children to process.
Task Scheduler	This graph on the Behavior Workview shows the execution flow of all scheduled behaviors, as well as each behaviors inputs. This allows the user to both see the execution order and to drag and drop objects onto specific inputs.

Task Script	A Task script contains a Task subroutine. The task subroutine must take one parameter of the appropriate type. Similar to Stand-Alone scripts, a Task script can contain any number of other routines, but must have one and only one Task subroutine. Each object in your simulation has a task list. You implement the behavior in a Task script by adding that script to the task list of one or more objects, and then running the simulation. When you run the simulation, Task scripts are executed every frame for every object to which the Task script is attached.
Tessellation	Refers to the manner in which the surface of a geometric object is modeled via polygons. Finer tessellations usually require the use of more polygons than a rough tessellation. For example, a cone that was tessellated using 100 polygons would, when rendered, appear much superior to a cone that was tessellated using only 10 polygons, since the 10 polygon tessellated cone would appear very faceted.
Texels	A contraction of “Texture element”, it refers to the individual texture elements of a texture image.
Text Fields	A text field user interface object is a simple object that allows a user to enter text using the keyboard. Text field objects are normally used as a single line data entry field. It gives the user text editing capabilities and also provides the point and click functionality expected of GUI applications.
Texture	A bitmap image usually created for the purpose of applying complex images to simple polygons to increase the visual quality of the simulation and to also improve the performance of the simulation.
Texture Draping	The process of applying a texture bitmap image stored in a file to a polygon or an entire geometry.
Texture Mapping	The process of applying a digitized image onto a polygon or structure composed of polygons.
Texture Tiling	Mechanism by which a texture image can be applied to a polygon in a manner such that the image is repeated a number of times horizontally across the polygon and/or vertically across the polygon, producing a ‘tiling’ effect.
Texture uv Coordinates	WorldUp allows you to specify how the texture is mapped onto a polygon, by allowing you to specify texture (uv) coordinates in polygon definitions.
The Universe	The container object for all global properties used by WorldUp.
TPS	Triangles per second. A commonly used statistic used to compare performance characteristics of graphics hardware.
Transformation Matrix	See Matrix.
Translation	A change in an object’s position.
Translucency	A material property that represents the extent to which the color value of a pixel is combined with the color value behind it, giving the affect of a transparent surface.

Traversal Order	The order in which nodes in a scene graph are processed while the simulation is running. WorldUp starts at the root node and processes the scene graph tree from top to bottom and left to right.
Type Workview	Type Workview is used for creating custom subtypes, viewing and changing type and object properties. The Type Workview consists of a Type Graph pane and a Property pane below it. The Type graph pane displays the complete set of worldup Types, including non-Node types such as Windows, the Universe, Sensors and MotionLinks.
Unit	An arbitrary measurement that you use to represent distance (inches, feet, centimeters, meters, etc.).
Universal Resource Locator (URL)	String properties used in VRML files to specify a file location and file name that contains data to be imported.
Universe or UP Files	A .UP file is a WorldUp universe file which is used to save your simulation. An .UP file, as opposed to an .WUP file, does not incorporate script, model, image, and sound file data into the .UP file. Instead a .UP file references other files which contain this information.
Upstream	An ancestor to a Behavior is said to be upstream.
User Script	The script that is run each time you click the User-Defined Action button. User scripts are useful for performing routine actions like resetting objects to their initial positions. The User script for a project is indicated by the User Script property.
Vertex	A single point in three-dimensional space, which defines a corner of a polygon. A sequence of vertices defines a polygon. Vertex properties include position, normal, and texture UV coordinates.
Vertex Normals	A direction vector used for shading and rendering. You can generate vertex normals with a modeling program. When WorldUp reads a vertex with a normal associated with it, it automatically renders the associated polygon as Gouraud-shaded. See also Normals.
Viewpoint	Defines the position and orientation from which the graphical universe is projected to the computer screen and rendered within a window. Each window has a viewpoint associated with it. The viewpoint represents the point of view of the observer.
Viewports	Viewports represent the actual drawing area of a window. All WorldUp windows have a default viewport assigned to them. This viewport is given the same dimensions as the client area of the window, but can be changed by the user.
VRML	An acronym for Virtual Reality Modeling Language. A specification for the design and implementation of a platform-independent language for virtual reality scene description.
Wireframe	A rendering setting in which textures, materials, and shading is not visible because only the outlines of polygons will be rendered, i.e. polygons will not be solid-filled. Rendering using the wireframe style typically achieves the highest frame rate of any of the rendering settings.

Workview	Provides a more efficient workflow by simplifying the User Interface, using tabs, for the task on which one is focused.
World Coordinate System	The World Coordinate System (WCS) originates at the center of the universe, defined as XYZ coordinates 0,0,0.
WorldToolKit	Sense8 Corporation also produces WorldToolKit, which is a C/C++ library used by programmers to build real-time 3D virtual reality applications. WorldUp is built on top of WorldToolKit.
WorldUp Player	The WorldUp players allow end-users to run your simulations without having to install WorldUp on their machines. There are both commercial and non-commercial players available. The non-commercial players are freely-distributable.
WRL	The VRML file format used for representing hierarchical 3D geometry and other data.
WUP Files	A .WUP file is a WorldUp project file that contains all the information required to run your simulation. A .WUP file, as opposed to an .UP file, does not reference any other files and is therefore suitable for distribution over the Internet so that others can run the simulation you built and exported as a .WUP file.
Yaw	The orientation of an object about the Y axis.
Yon Clipping Plane	The physical range in front of the viewpoint, beyond which objects are not rendered in that window. That is, objects appearing beyond the yon clipping plane are not rendered. Objects are rendered only in the area between the hither clipping plane and the yon clipping plane. See Hither Clipping Plane.
Z-buffer	A software or hardware buffer that stores Z coordinate information when rendering 3D scenes.