CONFIGTOOLKIT

USER MANUAL

V 1.0

CONTENTS

Managing configurations 2
Adding a new configuration 2
Editing a configuration 3
Configuration properties 3
Adding and deleting scenes from the configuration 4
Adding a scene 4
Deleting a scene 4
Disabling a scene 4
Save a configuration 4
Activate a configuration 5
Build a configuration 5
Delete a configuration 5
Configure GameObjects depending on the configuration 6
Overview 6
Setting up the layout: The FastConfig window 6
Example of general use: Configure Local Position 7
Specific configuration tools 8
Configure Activation 8
Configure Position 9
Configure Local Scale 9
Configure GUI Text 10
Configure GUITexture 10
Configure Material 11
Configure MeshFilter 11
Configure Text Mesh 12
Building configurations from command line 13
How to build from command line 13

MANAGING CONFIGURATIONS

To start defining and managing configurations, you must open the configuration window on "Edit/ConfigurationToolkit/Settings", and the configuration window will appear.

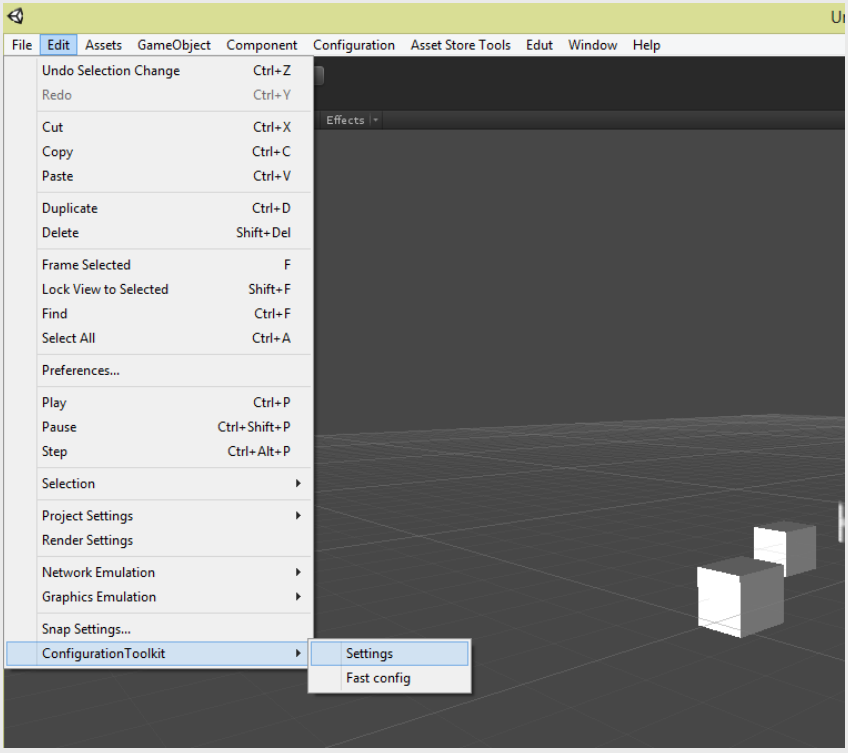


Image 1: Opening the configurations window

The configuration window is where you will manage all the configurations of your project.

ADDING A NEW CONFIGURATION

To add a new configuration, press the '+' button in the configuration window. The edit configuration window will show.

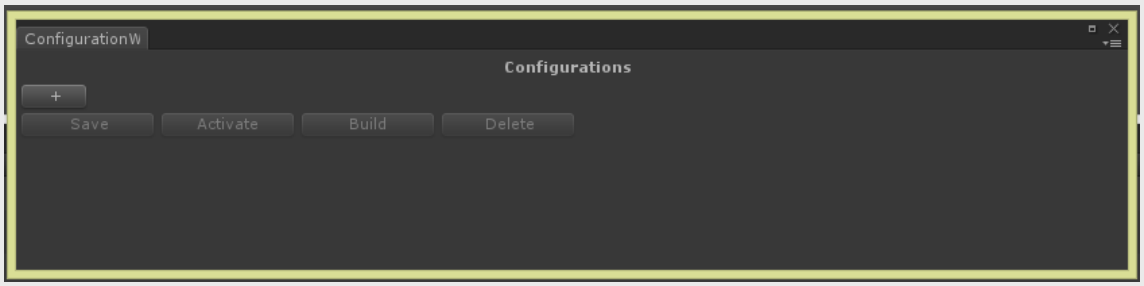


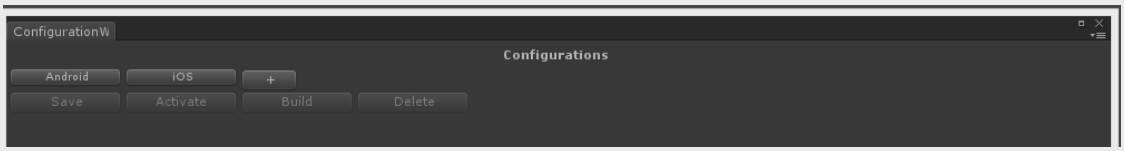
Image 2: Configuration window

Image 3: Edit configuration window

By default, the icons when you create a new configuration are the icons defined in the Build Settings.

EDITING A CONFIGURATION

To edit a configuration, select its name in the top bar of the Configuration window:



When you select a configuration, the configuration window will show up (as seen in Image 3: Edit configuration window) and the configuration name will change its color to yellow.

CONFIGURATION PROPERTIES

The fields in the “Edit Configuration” window are described below:

Config name	The name of the configuration, used to identify your configurations on ConfigurationToolkit. This name is used only by the ConfigurationToolkit, and to identify the configuration when building from command line.
App Name	This sets the field “Product Name” in PlayerSettings
Plattform	The target platform to build to. When the platform can have different targets, a second drop-down list will show next to this, to select the specific build target.
Icon	The icons to use with this configuration. By default, the icons defined in PlayerSettings are put here.
Scenes	The list of scenes on this configuration. For adding/removing scenes see Adding and deleting scenes to the configuration.

Add Scene	Field to add a new scene to the configuration.
Compiler definitions	List of definitions so you can make configuration dependent code with #if, #elif and #else.

ADDING AND DELETING SCENES FROM THE CONFIGURATION

ADDING A SCENE

To add a scene to the configuration, select it from the “Add Scene” field of the configuration window.

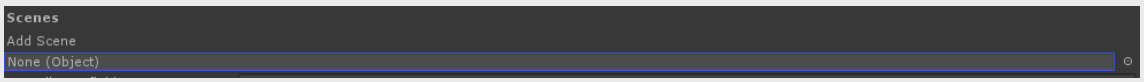


Image 4: Add scene field without scene selected

Once selected, a new “+” button will appear below the “Add scene field”. Press it to add the selected scene to the current config.

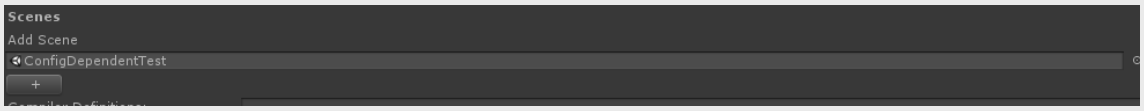


Image 5: Add scene field with a scene selected

Warning! If you don´t press the “+” button, the scene will not be added to the configuration.

Once you have added the scene, it will appear in the scenes list as shown below.

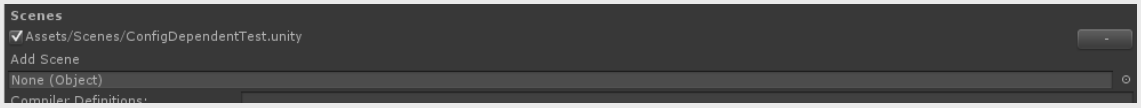


Image 6: Scenes list with a scene added

DELETING A SCENE

To delete a scene from the configuration, press the “-“ button located at the right of the scene name in the “Scenes” list.

DISABLING A SCENE

From the config manager, you can disable a scene without removing it from the list. Simply click the checkbox at the left of the scene name to disable it. The scene will not be added to the build.

SAVE A CONFIGURATION

Once you have defined a configuration you must save it. When you create or edit a configuration, a message will be shown at the bottom left corner of the configuration window warning you that the configuration is not saved.

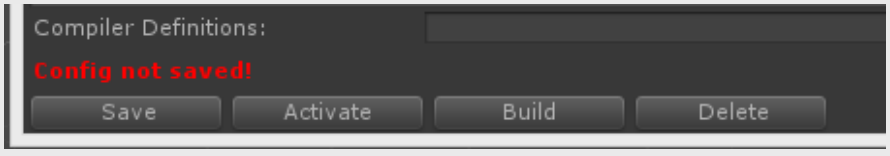


Image 7: "Config not saved" message

To save the configuration, simply press the “Save” button. The “Config not saved!” message will disappear.

The first time you use the toolkit, a new file called “config.txt” will be created on “Assets/Resources/”. This is where we’ll save all your configurations.

ACTIVATE A CONFIGURATION

To apply all your configuration settings to the project, you must activate a configuration.

To activate a configuration it must be selected, by pressing the configuration name on top of the configurations window. Once the configuration name is highlighted in yellow, you can activate it by pressing the “Activate” button. When you activate a configuration, the target platform for Unity build is also changed.

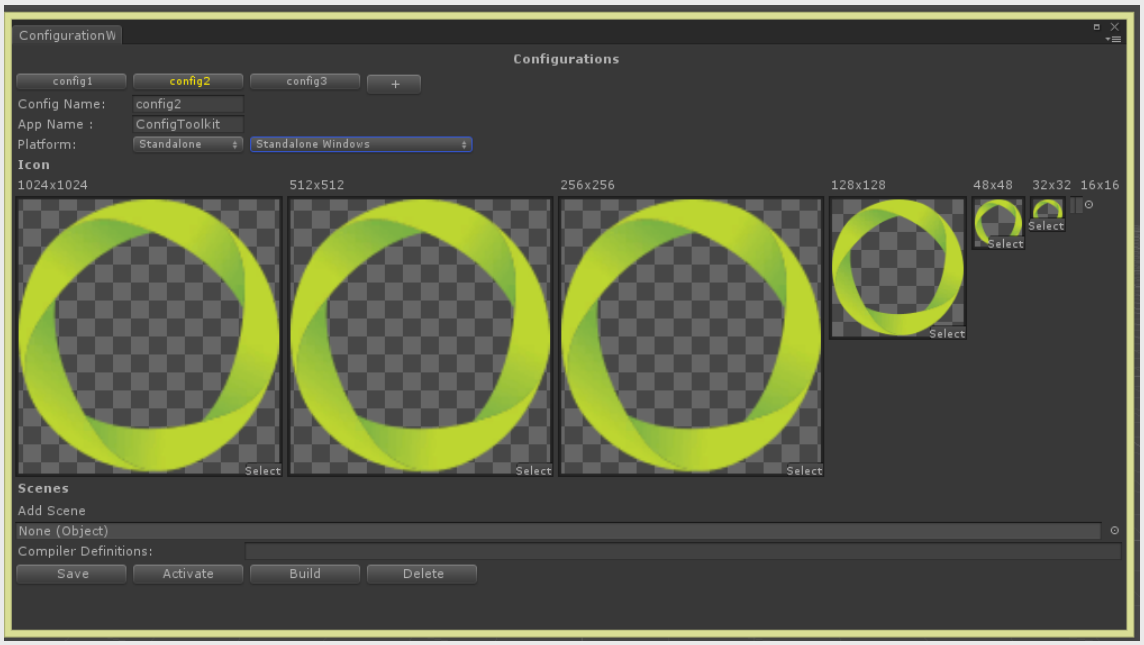


Image 8: The configuration name will turn yellow when it's selected.

A faster mode of activating a configuration is by using the FastConfig window.

BUILD A CONFIGURATION

To build a configuration it must be selected, by pressing the configuration name on top of the configurations window. Once the configuration name is highlighted in yellow, you can build it by pressing the “Build” button.

DELETE A CONFIGURATION

To delete a configuration it must be selected, by pressing the configuration name on top of the configurations window. Once the configuration name is highlighted in yellow, you can delete it by pressing the “Delete” button.

If you are using a “Configure XXX script” with the configuration you’ve deleted, the setting for that configuration will be removed **only on the active scene**. In other scenes, it will take the next configuration on the list. You’ll have to manually remove the settings for the deleted configuration from your “Configure” scripts that use this configuration in other scenes.

CONFIGURE GAMEOBJECTS DEPENDING ON THE CONFIGURATION

With ConfigurationToolkit comes a sample scene called “ConfigurationToolkitSample”. This scene contains examples about different tools to configure GameObjects.

OVERVIEW

The “Configure GameObject” tools allows you to set specific values for each configuration to a GameObject. With one of these scripts attached, you have to set values for each configuration you need. If no values are set for a specific configuration, the default config in the GameObject will be taken.

SETTING UP THE LAYOUT: THE FASTCONFIG WINDOW

To quickly switch between active configurations without opening the Configuration Window, you can use the FastConfig window.

The window is located on “Edit\ConfigurationToolkit\Fast Config”. This opens a small window with a drop-down list of the defined configs.

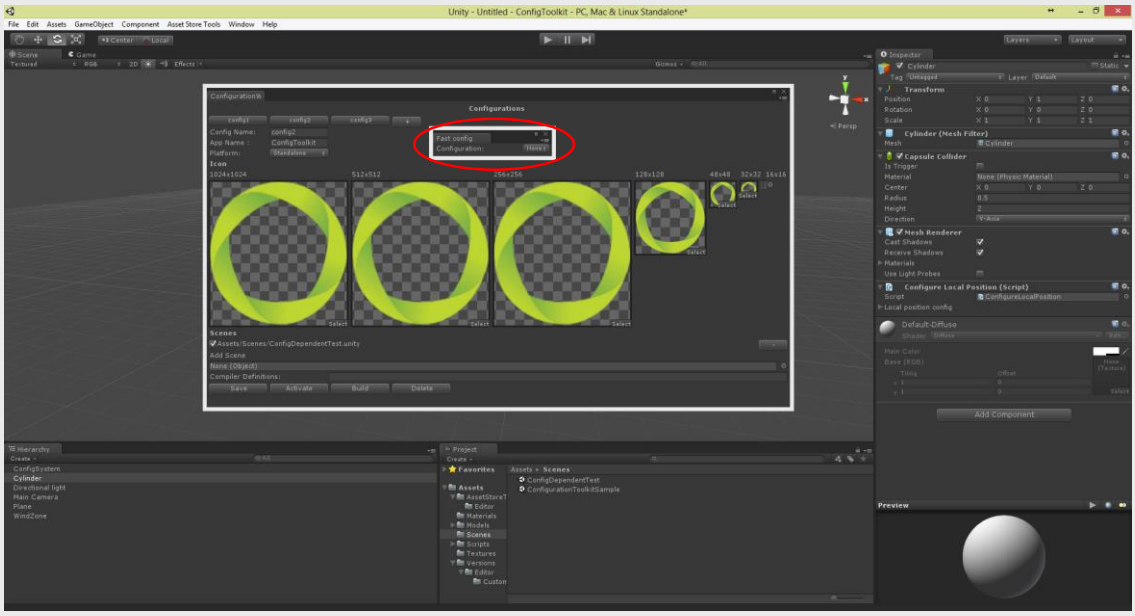


Image 9: The "Fast Config" window (highlighted in red)

We usually dock this small window on the bottom-right corner of the unity editor, to have a quick access to the tool.

Using this window **doesn’t perform the Switch Platform action** because of the time it can consume. It changes the **icons for the configuration platform**, the **scripting symbols for all the platforms** (so dependent compilation could be done) and the **product name**.

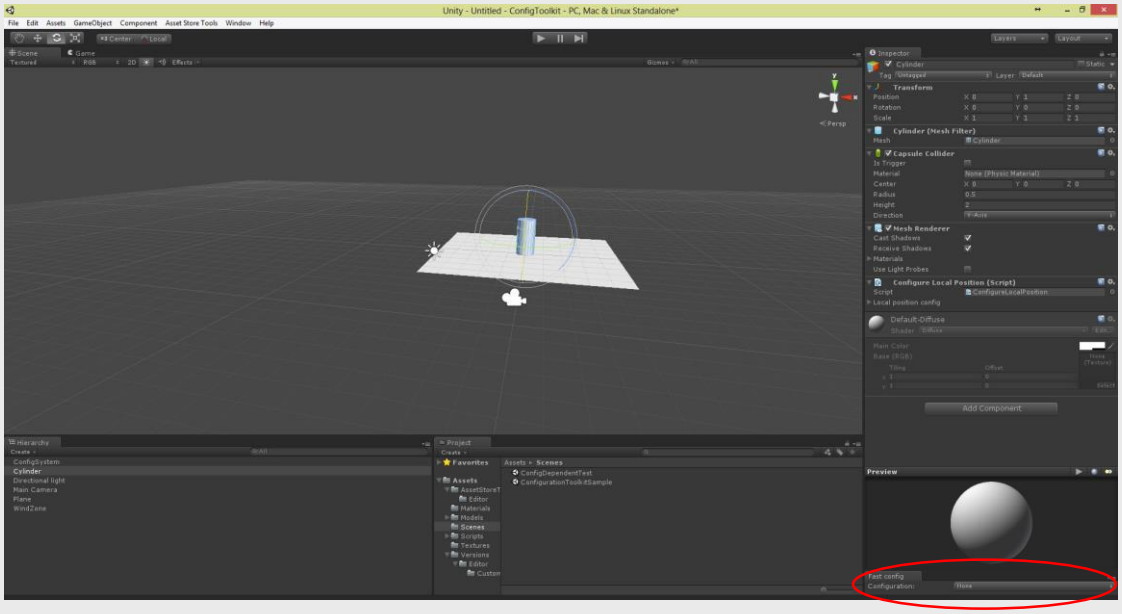


Image 10: Fast Config window docked

To quickly switch between configurations, simply select it from the drop-down list of the FastConfig window.

EXAMPLE OF GENERAL USE: CONFIGURE LOCAL POSITION

The basic principles of setting up values for different configuration applies to all the scripts. To show how we can add specific values for a configuration, we'll take for example the "ConfigureLocalPosition" script.

First, we'll define three configurations called config1, config2 and config3, as described in Adding a new configuration. Then we'll add a GameObject (cylinder in this example) to the scene, and attach out ConfigureLocalPosition script.

When attached, the values in the inspector will look like this:

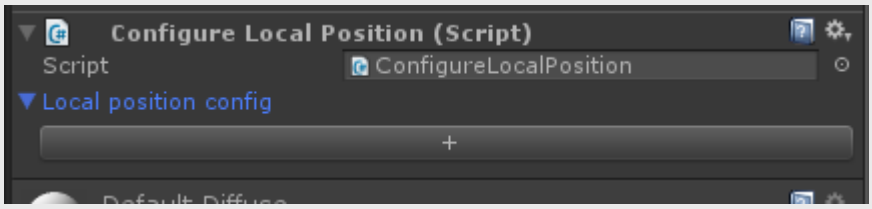


Image 11: Configure Local Position script after attachment to GameObject

If we press play, the cylinder will be placed in the same position we have defined in the Transform field of the Inspector. **If the active config is not set in the script, the GameObject will take the values set in the Transform component of the Inspector**.

To set a specific value for a configuration, we'll press the "+" button, and the ConfigureLocalPosition fields will change into this:

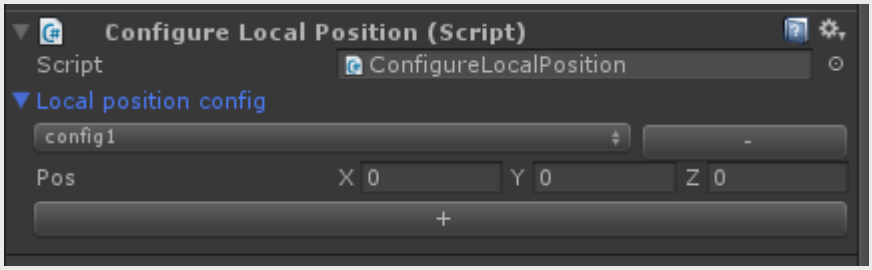


Image 12: Adding a specific value for a config in ConfigureLocalPosition

Here we can define the position for one of the configurations we have defined. I'll configure a position for "config2" configuration selecting it from the drop-down list, and I'll add a new specific position for "config3" by pressing the "+" button and selecing it from the list. To remove a configuration you have to press the "-" button at the right of the configuration name.

The Script values should be like this:

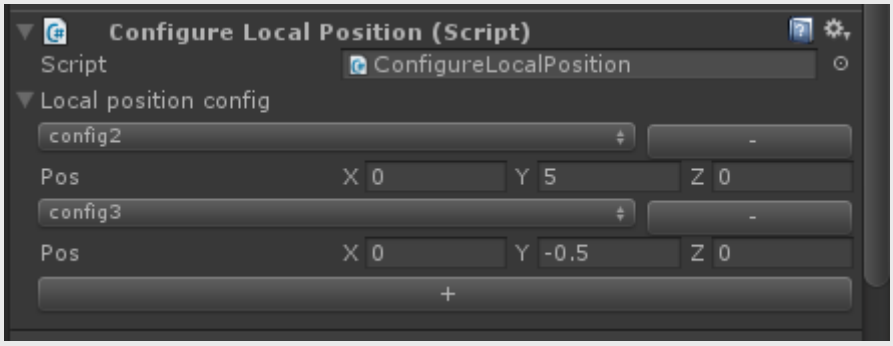


Image 13: Configure Local Position script for two configurations

Now, if I set the "config2" configuration as active (either using the FastConfig window or the Activate button) and press Play, my GameObject will have a position of (0, 5, 0). If I set the "config3" as active and press Play, my GameObject will have a position of (0, 0.5, 0), and if I set any other config as active and press Play, the GameObject will have the position defined in the Transform component of the GameObject.

Note that the position only changes when pressing play, not when switching the active config. In the Editor, the position of the object will be defined by its Transform. **ConfigToolkit tools only affect the GameObject's properties on Run time.**

SPECIFIC CONFIGURATION TOOLS

The "ConfigurationToolkitSample" scene contains an empty GameObject named "ExampleGameObject" with one child for each Specific Configuration tool that ConfigToolkit has, showing the use them.

Below is a description of how each script works. To see a guide step-by-step on how to use this scripts, go to Example of general use: Configure Local Position.

CONFIGURE ACTIVATION

This script is used to enable/disable a GameObject depending on the selected configuration.

The value for each configuration is set with the checkbox labelled "Active":

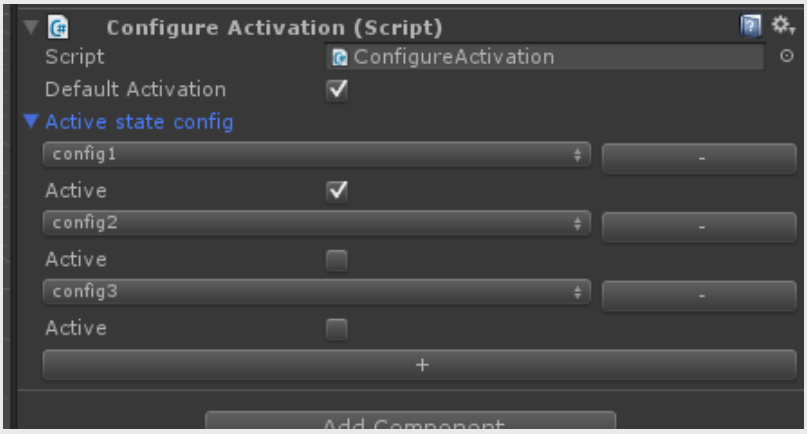


Image 14: Configure Activation script

This script has a default value, “Default State” that will be applied if a configuration is not set in the script. In the above example, for config1 the GameObject will be enabled, disabled for config2 and enabled for config3.

CONFIGURE POSITION

This script allows you to set a position for a specific configuration. If no position is set, the GameObject will take the values set on the Transform component.

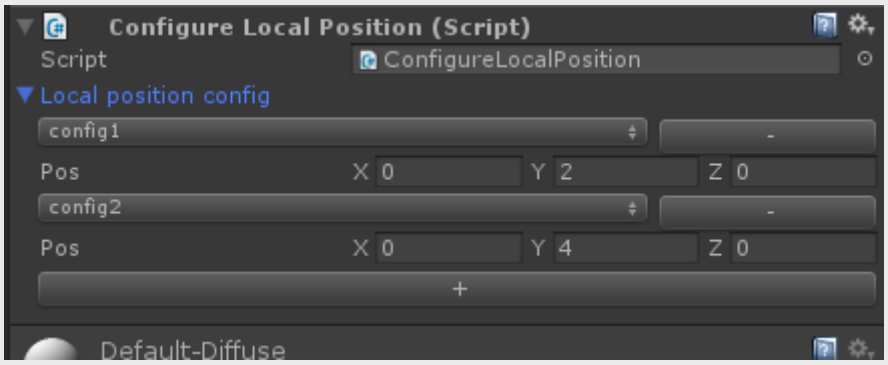


Image 15: Configure Local Position script

In the above example, the GameObject will have a position of (0, 2, 0) for config1, (0, 4, 0) for config2, and will not change its position for config3.

CONFIGURE LOCAL SCALE

This script allows you to configure the local scale of a GameObject.

Image 16: Configure Local Scale script

CONFIGURE GUI TEXT

This script changes the Text value of a GUIText for each configuration defined. If a configuration is not set in the script, the value will be the same as in the GUIText component.

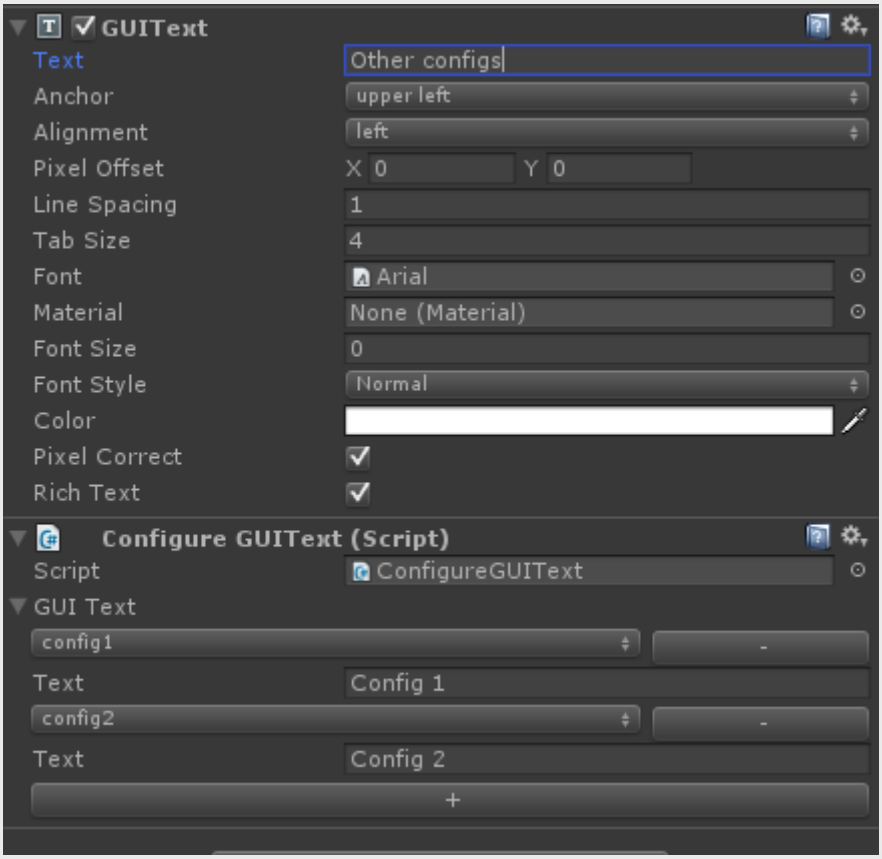


Image 17: Configure GUIText script

In the example above, the text for config1 will be “Config 1”, for config2 will be “Config 2” and for config3 and any other config will be “Other configs”.

CONFIGURE GUITEXTURE

This allows you to change the texture image from a GUITexture component for each configuration.

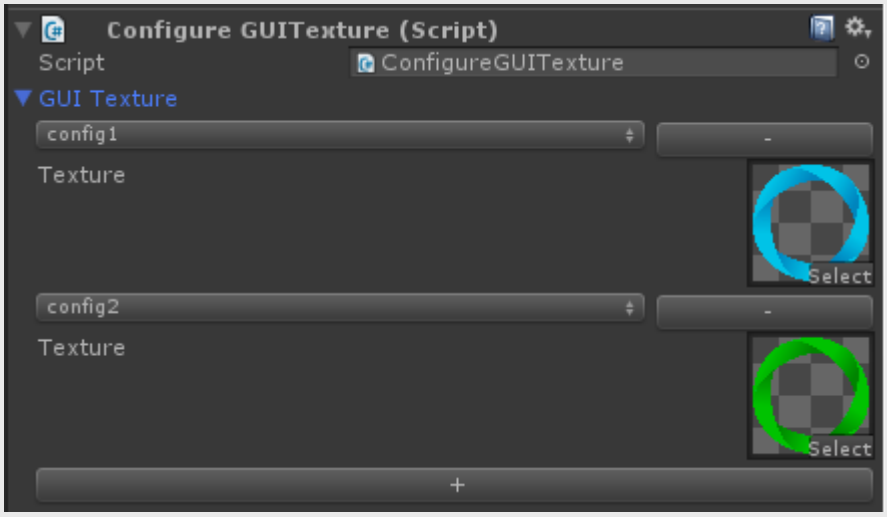


Image 18: Configure GUITexture script

CONFIGURE MATERIAL

This allows you to define a Mesh Renderer's material for each configuration. Multiple materials are allowed.

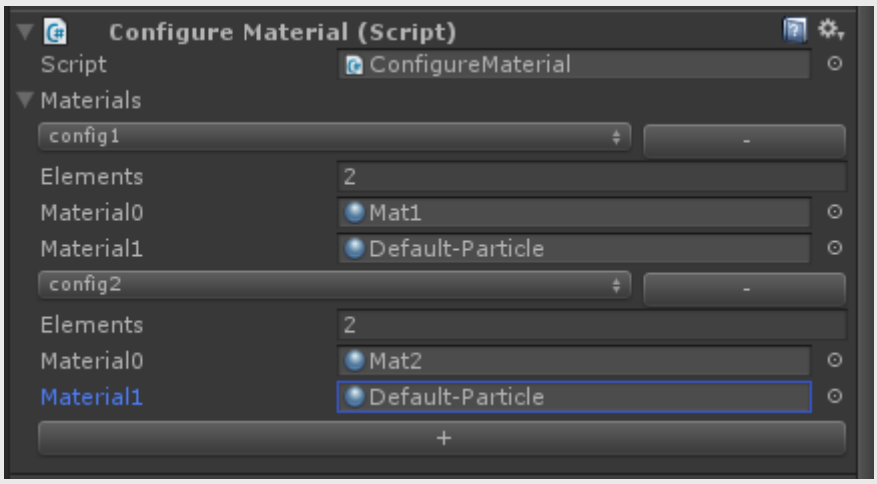


Image 19: Configure Material script

CONFIGURE MESHFILTER

This script allows you to define a mesh from a MeshFilter component for each configuration. Combined with Configure Material, it's a simple way of having different meshes for each platform (IE low poly for mobile plattforms and hi-poly for PC).

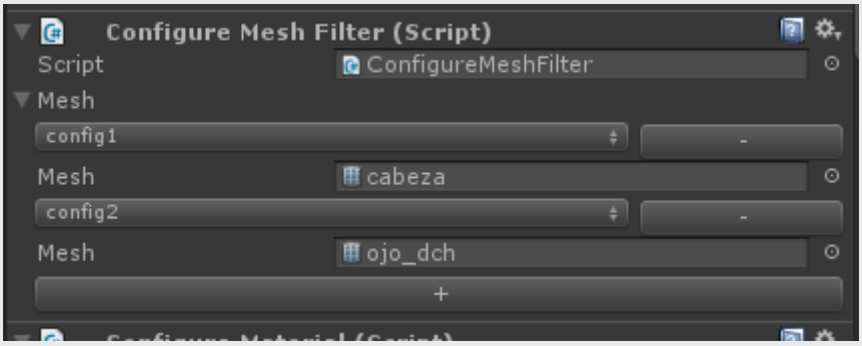


Image 20: Configure MeshFilter script

CONFIGURE TEXT MESH

Similar to Configure GUI Text, this script allows you to define the Text value of a TextMesh for each configuration defined.

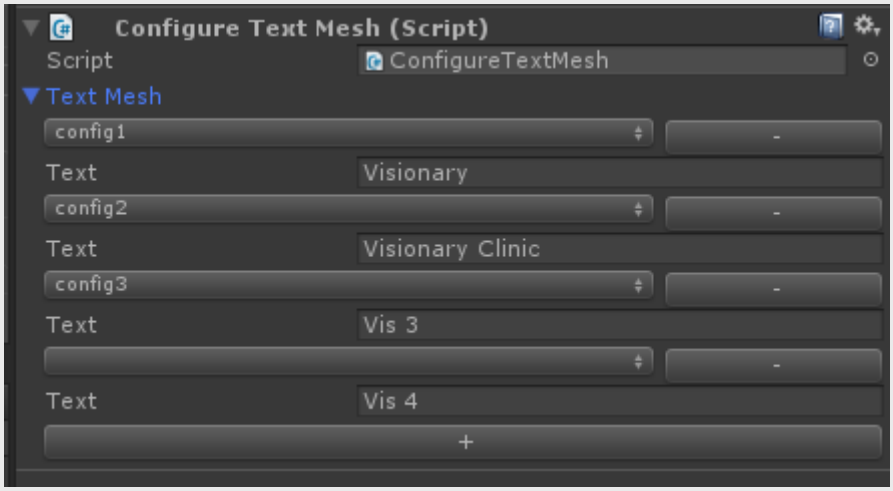


Image 21: Configure TextMesh script

BUILDING CONFIGURATIONS FROM COMMAND LINE

To use this feature you need to have Unity Pro, for the target platform you want to build to.

You can build a configuration you've defined with Configuration Toolkit from command line. This comes handy when building a Continuous Integration Server.

To build from command line, you have to user the CommandLineBuild class. You have an overview of calling scripts from command line here.

HOW TO BUILD FROM COMMAND LINE

First, you have to have at least a configuration defined as seen in Adding a new configuration.

Then you can close Unity and launch a build from command line. To do so, you'll have to write a command like the following (works on DOS window and .bat file):

```
"D:\Unity43\Editor\Unity.exe" -projectPath "D:\ConfigToolkitTest" -executeMethod CommandLineBuild.Build -config "Win2" -buildPath "D:\ConfigToolkitTest\Binary\Win2\TestWin2.exe" -quit -batchmode -logfile "win2log.txt"
```

Let's break down the previous line and explain the arguments.

"D:\Unity43\Editor\Unity.exe" is the path to my Unity executable file. It's usually located on the "Program Files" folder.

-projectPath "D:\ConfigToolkitTest" sets the path to the root my project. You have to put the path to your project

-executeMethod CommandLineBuild.Build Calls the Build method of the CommandLineBuild class

The following two arguments must be right next to –executeMethod:

-config "Win2" This is the name of your config as defined on Config Toolkit.

-buildPath "D:\ConfigToolkitTest\Binary\Win2\TestWin2.exe" This sets the output file/path for the build. If you were developing for Android, you should put a path to an APK for example, for iOS it sould be a path, etc.

-quit Tells Unity to close after build. If you remove this parameter, it will remain opened after the build.

-batchmode Tells Unity to run in batchmode (no popups or dialogs)

-logfile "win2log.txt" Saves a build log to the specified file. Useful to locate build errors.