



C o m m u n i t y E x p e r i e n c e D i s t i l l e d

Extending Unity with Editor Scripting

Put Unity to use for your video games by creating your own custom tools with editor scripting

*Foreword by John P. Doran, Technical Game Designer
Author of Unity Game Development Blueprints and Mastering UDK
Game Development*

Angelo Tadres

[PACKT]
PUBLISHING

Extending Unity with Editor Scripting

Put Unity to use for your video games by creating your own custom tools with editor scripting

Angelo Tadres



BIRMINGHAM - MUMBAI

Extending Unity with Editor Scripting

Copyright © 2015 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: September 2015

Production reference: 1150915

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78528-185-3

www.packtpub.com

Credits

Author

Angelo Tadres

Project Coordinator

Kinjal Bari

Reviewers

J. Alberto Gandullo Avila

Jeremy Jones

Noah Johnson

Fernando Matarrubia

Hugo Ruivo

Eric Spevacek

Proofreader

Safis Editing

Indexer

Mariammal Chettiyar

Graphics

Jason Monteiro

Commissioning Editor

Veena Pagare

Production Coordinator

Conidon Miranda

Acquisition Editor

Sonali Vernekar

Cover Work

Conidon Miranda

Content Development Editor

Riddhi Tuljapurkar

Technical Editor

Vivek Pala

Copy Editor

Pranjali Chury

Foreword

While perhaps not as glamorous a job as being a gameplay programmer, a tools programmer can make your game development experience much more enjoyable. They truly are the unsung heroes of game development. In fact, AAA studios heavily rely on using tools to make aspects of game development easier to use for designers and artists. Tools also help to reduce tediousness in the creation of content for game projects.

While these tools were often created as separate programs to be run in conjunction with the game engine in the past, one of the things I love about working with the Unity game engine is the fact that with some fairly trivial scripting, you can extend the editor. This allows users to tailor the editor to suit their project's needs and requirements. Additionally, just as Unity was originally created for a game project but grew into a lot more, the custom tools readers will go on to create applications that have the possibility to be extraordinarily successful on Unity's Asset Store, much like NGUI, Playmaker, ProBuilder, and UFPS.

Since I started working with Unity in 2007, I have worked with a lot of tools and have done a fair bit of tools programming personally. While creating my own tools, I often needed to do extensive external research and come up with a lot of things on my own because most of the necessary information was not documented well. I am exuberant that someone has compiled the majority of this information into one place.

Over the course of this book, you will see how you can create your own custom tools starting with simple ones such as gizmos, then moving on to customize the Inspector for the different components you add, and learning how to create your very own Windows with their own custom GUI. Angelo has broken down the concepts and has made it quite easy to see when you would want to use these tools. Throughout this book, he shows practical examples of when you would want to use these particular features from their inception to getting published on the Asset Store. He has also included additional tips and tricks along the way, such as how to set up Git, easily make multiple builds of your projects, as well as get your project up on mobile devices in a flash.

Reading Angelo's work, I am not surprised by the range of content covered in this book. His work as a lead engineer for DeNA as well as his strong technical background, no doubt, gave him the knowledge needed to get this book out to the world. The breadth of content included in this book will give you a strong foundation on which you can build your own tools.

Gifted tools programmers can make all the difference in the world of game projects. This book provides a roadmap on how you can get there.

John P. Doran

Technical Game Designer

Author of Unity Game Development Blueprints and Mastering UDK Game Development

About the Author

Angelo Tadres is a Chilean software engineer, living the dream of working in the mobile video game industry.

Hailing from Santiago, Chile, he began his career doing research and development for video games and applications that are designed to assist the blind and visually impaired with their orientation and mobility skills. After passing quickly through the telecommunications industry – working with value-added services and mobile applications – he got the opportunity to join the Santiago studio of DeNA, one of the world's largest mobile video game companies.

In 2013, Angelo was asked to move to Vancouver, Canada, to become a lead software engineer, where he helped build the fledgling Canadian studio and, in particular, championed Unity 3D, paving the way for other teams' adoption and use.

He's known for getting things done by shooting first and asking questions later. When he is not coding and pushing content to GitHub, you'll find him playing table tennis with his friends or running along the sea wall. To know more about him, visit his website at <http://angelotadres.com/>.

This book is dedicated to my daughter, Antonia Tadres, and my wife, María Jose Arcos, the person whom I love and who has always supported me in all my crazy projects, including the time when I said "You know what? I want to write a book!"

Thanks to my mom; dad; my whole family; and my friends Jorge Bravo and Vartan Ishanoglu for always being there to push me whenever I doubted myself.

I would also like to say thanks to all the people who work at DeNA Studios Canada for making the past 2 years the most amazing ones of my life.

Finally, I would like to thank the Packt Publishing staff for their assistance through the process and the technical reviewers for their feedback, especially Riddhi Tuljapurkar and Fernando Matarrubia.

About the Reviewers

J. Alberto Gandullo Avila graduated from the University of Seville after a 5-year course in computer science (BA/MA). After this, he worked in Seville as a software developer in the field of enterprise management tools for more than 3 years. However, he always liked other fields such as computer graphics and mobile software more, so he began to self-train in the development of mobile apps and mobile games, specifically in the new technology of augmented reality; this was his first contact with the Unity game engine. Thanks to his proficiency in this field, in 2013, he was hired in London (UK) by a small start-up dedicated to the development of educational video games for mobile devices based on augmented reality. At this stage, he became an expert in developing games using technologies such as C# and Unity. After one and a half years in London, Alberto was hired in Bangkok (Thailand) by a company dedicated to developing F2P games for mobile devices.

Jeremy Jones is a game developer who graduated from Neumont University and has a passion for making robust systems within games. He has created many games and several tools in his own game engine and Unity. In his free time, he likes to go hiking, work out, and draw road designs.

I would like to thank my friends at Neumont University for their support and my family on the East Coast for always believing in me.

Noah Johnson is a technical artist currently working at InContext Solutions. He specializes in pipeline tools and extensions between Unity, Maya, and standalone Python apps. He teaches game engine scripting courses as an adjunct professor at Columbia College Chicago and is currently working on an independent Unreal 4 horror game project. His background in game system scripting and 3D asset creation has dovetailed into a skill set that focuses on tools that make content creation simpler and easier to iterate.

Fernando Matarrubia is a passionate traveler and game maker. He completed his bachelor's degree in computer engineering, for which he was required to travel between three cities and two countries. After that, he got a master's degree in video game development from the Complutense University of Madrid. He has been working with Unity for almost 6 years and loves to create fun pieces of entertainment. He has participated in several titles for platforms such as PS3, PC, Mac, and mobile devices.

Fernando is currently living with his wife and working as a software engineer in the San Francisco Bay Area.

Hugo Ruivo is a self-taught game programmer, who is currently making games for both the mobile and desktop platforms. Alongside games, he also creates tools that help him and his team in the making of their products. He even launched one of his own tools for Unity 3D on the marketplace, Achievement Service Manager.

Ever since he found out how games were made, he couldn't stop learning about the many disciplines of game development, trying to make his own engine, learning new frameworks and technologies, and specializing in some of the best game engines in the industry, such as Unity 3D and UE4.

I would like to thank to my brother and my best friends, who have always given me the inspiration and strength to keep moving forward. I would also like to thank Packt Publishing and Angelo Tadres for the opportunity to contribute to this book and, at some point, to be able to help others learn the same way as I have been learning.

Eric Spevacek, once an independent developer in Chicago, is now an industry technical artist based out of Southern California. His holistic approach to game development and independent experience have helped guide and shape his work in tool development. At work, he is responsible for the creation and maintenance of content creation tools with an emphasis on user experience and streamlined modern workflows. The current trends of accessible commercial game engines and their long-term impact on the industry excite him.

www.PacktPub.com

Support files, eBooks, discount offers, and more

For support files and downloads related to your book, please visit www.PacktPub.com.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<https://www2.packtpub.com/books/subscription/packtlib>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Free access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view 9 entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface	vii
Chapter 1: Getting Started with Editor Scripting	1
Overview	2
Editor scripting basics	3
What is an editor script?	3
The Editor folder	5
Introducing Run & Jump	7
Playing the video game	8
Creating a new level	9
The Level Creator tool	14
Defining the chapter goals	15
Preparing the environment	16
Performing automation	17
Summary	20
Chapter 2: Using Gizmos in the Scene View	21
Overview	22
Defining the chapter goals	23
Creating gizmos through code	24
The OnDrawGizmos and OnDrawGizmosSelected methods	24
Adding gizmos using the DrawGizmo attribute	27
The Gizmos class	30
DrawCube	30
DrawWireCube	31
DrawSphere	32
DrawWireSphere	32
DrawRay	33
DrawLine	34
DrawIcon	34
DrawGUITexture	35
DrawFrustrum	36

Adding a structure to our levels	37
Implementing the gizmo grid	38
Implementing the snap to grid behaviour	44
Summary	47
Chapter 3: Creating Custom Inspectors	49
Overview	50
Defining the chapter goals	51
Upgrading the Level class	52
Understanding how an inspector works	53
Creating a custom inspector	55
Using the CustomEditor attribute	55
Playing with the inspector message methods and target variable	56
Adding the GUI elements	58
Implementing the resize feature	61
Using buttons to trigger actions	62
Working with layouts	66
Creating complex layouts	67
Improving the inspector without custom inspectors	70
What is a Property Drawer?	70
Built-in Property Drawers	71
Range	71
Multiline	72
TextArea	73
ContextMenu	73
ContextMenuItem	74
Built-in Decorator Drawers	75
Header	75
Space	76
Tooltip	76
Creating you own Property Drawers	77
Using drawers inside a custom inspector	81
Using SerializedObject and SerializedProperty	82
Summary	84
Chapter 4: Creating Editor Windows	85
Overview	86
Defining the chapter goals	87
Creating the base for an editor window	87
Using the EditorWindow class	88
Playing with the EditorWindow message methods	89
Using Hotkeys to trigger menu items	91

Implementing the Palette	92
Creating a category system	93
Finding assets using the AssetDatabase class	95
Implementing the GUI for the Palette	97
Creating tabs	97
Creating a scrollable area	99
Integrating the Palette with the Level Creator tool	105
Creating an event	105
Subscribing to an event	106
Summary	111
Chapter 5: Customizing the Scene View	113
Overview	114
Defining the chapter goals	114
Defining the Editor modes	116
Customizing the Scene View	117
Using the OnSceneGUI message method	117
Playing with the Scene View tools	119
Controlling the focus over our game objects	121
Detecting Scene View events	122
Getting the mouse position	123
Capturing mouse events	126
Implementing the Level Creator modes	128
The View mode	128
The Paint mode	129
The Erase mode	131
The Edit mode	132
Using the Handles class	136
Adding the final details to Level Creator	140
Using hiding flags	140
Summary	143
Chapter 6: Changing the Look and Feel of the Editor with GUI Styles and GUI Skins	145
Overview	146
Defining the chapter goals	146
Changing the look and feel of the Level Creator tool	147
Using GUIStyles in our GUI components	147
Working with the GUIStyleState instances	152

Changing the look and feel using a simpler approach	156
Creating a GUISkin asset	156
Integrating and using a GUISkin	159
Summary	162
Chapter 7: Saving Data in a Persistent Way with Scriptable Objects	163
Overview	163
Defining the chapter goals	164
Preparing the environment	164
Updatable gravity in levels	164
Playing with gravity	165
Implementing a Scriptable Object	166
Creating the data class	166
Generating an asset to contain the data class	167
Integrating the Scriptable Object with the level	171
Updating the Level and the LevelInspector class	171
Tweaking the level settings in the play mode	174
Summary	175
Chapter 8: Controlling the Import Pipeline Using AssetPostprocessor Scripts	177
Overview	178
Defining the chapter goals	178
Using the AssetPostprocessor class	178
Improving the import pipeline	181
Overwriting the background and level piece assets settings	181
Using a DLL file for the AssetPostprocessors	184
Creating and setting up a DLL project	185
Integrating the DLL file to the main project	189
Summary	192
Chapter 9: Improving the Build Pipeline	193
Overview	193
Defining the chapter goals	194
Preparing the environment	194
Automating the BuildPipeline class	194
Adjusting the player settings	195
Using the BuildPipeline class	196
Creating an editor window and learning about EditorPrefs to persist data	199
Adding version control to your project	204

Interacting with external scripts	206
Displaying the build information in the video game	206
Using the bash script in our pipeline	208
Distributing your video game using AppBlade	211
Creating an AppBlade account	212
Uploading the build	213
Summary	216
Chapter 10: Distributing Your Tools	217
Overview	217
Defining the chapter goals	218
Preparing the environment	218
Sharing code using a Unity Package	219
Creating a package	219
Importing a package	221
Sharing code using Git submodules	222
Creating a submodule	222
Using a submodule	223
Publishing in the Asset Store	225
Installing the Asset Store Tools	225
Becoming a publisher	227
Uploading the package	229
Using the Mass Labeler	232
Uploading and submitting the project	234
Summary	236
Index	239

Preface

Unity is a development platform for creating multiplatform 3D and 2D video games, which is adopted by several studios and indie developers who are looking for something simple, flexible, and powerful. One of its most interesting features is the extensible editor, allowing you to make Unity work for your video game using editor scripting.

If you are looking for a book that will show you how to deal with tasks that are beyond the implementation of Gameplay and are more related to automating and simplifying the creation of content, such as the assets that require a special configuration to make them usable in your levels, and how to enable pipelines to consume and create artifacts used by your video game, then this book is for you.

While improving the workflow of *Run & Jump*, a 2D platformer videogame, you will learn all the basics of editor scripting, creating an ad hoc tool that works as a level editor, customizing the way Unity imports assets, and getting control over the build creation process. As a bonus, you will also learn how to share the tools created inside your team or sell them at the Asset Store.

By the end of this book, you will be able to extend all the concepts that you learned to build your own tools and customize the Unity editor in future video game projects with confidence.

You can consider this as an entry point to make your development workflow easier.

Enjoy!

What this book covers

Chapter 1, Getting Started with Editor Scripting, introduces you to Unity editor scripting and explains why this is useful to improve the development workflow. In this chapter, the video game, *Run & Jump*, which is used as a base for this book is presented.

Chapter 2, Using Gizmos in the Scene View, explains how to use gizmos to display debug information in the Scene View. Here, we implement a grid with gizmos to be used as guides in the level editor.

Chapter 3, Creating Custom Inspectors, discusses how to improve the way the Unity components and scripts are presented in the inspector window, creating custom inspectors and using property and decorator drawers. In addition to this, you will learn how to start adding and using the editor GUI components. Here we go through the process of making a custom inspector for the class responsible for the level logic in *Run & Jump*.

Chapter 4, Creating Editor Windows, covers how to create an editor window to present information and interact with features in a custom tool. Using some of the editor GUI skills developed in the last chapter, we create a *Palette* window, which is a quick and visual way to access the prefabs used as building pieces for the video game levels, grouping them by categories.

Chapter 5, Customizing the Scene View, dives into how to add the editor GUI components directly to the Scene View and capture specific events to expand their capabilities. Step by step, we add GUI components to enable and disable different modes we are going to implement on the level editor, like View, Paint, Edit and Erase, changing the way how the user interacts with the tool.

Chapter 6, Changing the Look and Feel of the Editor with GUI Styles and GUI Skins, explains how to change the look and feel of the Unity editor custom tools. Here we finish the level editor investing our time modifying the appearance of it.

Chapter 7, Saving Data in a Persistent Way with Scriptable Objects, describes how to save data in Unity and manipulate it as a reusable asset using scriptable objects. We walk through the process of reallocate certain properties from the class responsible for the level logic to a scriptable object class, making them reusable across levels.

Chapter 8, Controlling the Import Pipeline Using Asset Postprocessor Scripts, demonstrates how to improve and control the importing pipeline using Asset Postprocessor scripts. We work in automating the process of changing the import settings of the assets imported to the project to make them usable by the video game in an easy way.

Chapter 9, Improving the Build Pipeline, discusses how to automate and improve the build creation pipeline modifying the Unity player settings through code and calling scripts outside Unity. Here, we create a basic build pipeline for Run & Jump that publishes the mobile version of it in a distribution platform called AppBlade.

Chapter 10, Distributing Your Tools, concludes this book by showing how to use Unity packages and Git submodules for custom tools distribution, suitable for sharing inside a team, and how to sell content on the Asset Store.

What you need for this book

To follow this book, you will need to download a copy of Unity available at <https://unity3d.com/get-unity>.

You can use any version of Unity from version 5.0, but we recommend the latest 5.x version, which at the time of writing this is version 5.1.2 (all screenshots have been updated to this version). Don't worry about the kind of license you have, the examples will work with the Personal and Professional Edition.

While working with this book, we will use as base project the video game *Run & Jump*, available at <https://github.com/angelotadres/RunAndJump>.

You must have the *Run & Jump* project in order to test the code in this book.

Who this book is for

This book is for anyone who has basic knowledge of Unity programming using C# and wants to learn how to extend and create custom tools using Unity Editor Scripting to improve the development workflow and make video game development easier.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Create a script called `LevelInspector.cs` inside the folder `Editor`"

A block of code is set as follows:

```
public override void OnInspectorGUI() {  
    DrawLevelDataGUI();  
    DrawLevelSizeGUI();  
}
```


When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:


```
public override void OnInspectorGUI() {  
    DrawLevelDataGUI();  
    DrawLevelSizeGUI();  
}
```

Any command-line input or output is written as follows:

```
$ git submodule update
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Select the category **Misc** and then click on the **Sign** piece"

 Warnings or important notes appear in a box like this.

 Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail feedback@packtpub.com, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files from your account at <http://www.packtpub.com> for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit <http://www.packtpub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/submit-errata>, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to <https://www.packtpub.com/books/content/support> and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at questions@packtpub.com, and we will do our best to address the problem.

1

Getting Started with Editor Scripting

Unity is a powerful engine that enables creative people like you to build video games in different platforms.

After developing a few projects on it, you will realize that each of these could have been a better experience if you'd had a tool at that time to help you in the creation of content for your video game or in the automation of all those manual repetitive tasks that always end up generating a problem at the worst moment just because of Murphy's Law.

To create tools based on your video game requirements, Unity provides an **editor scripting API** to do it in a quick and fully integrated way. However, the documentation available for building such tools by yourself is not the best.

The main aim of this book is to give you a tour of some of the most important topics about editor scripting . We are going to explore its API when at the same time we implement custom tools to improve the development workflow in *Run & Jump*, a 2D platformer video game.

In this chapter, we will cover the following topics:

- Basics of editor scripting
- *Run & Jump* presentation and definition of the scope of the custom tools

Overview

Probably, at this point, you are familiar with the basic concepts of Unity and we can safely assume that you know how to create a small video game from scratch without too many complications. You know, for projects of this size, almost everything is always under control and nothing takes too much time to be done. Basically, it is like a little paradise in the video game developer's land.

However, when the project starts increasing in size in terms of complexity, you will notice that certain tasks are repetitive or subject to error, generating a considerable amount of effort and waste of time. For example, the mechanics of your video game are quite unique and it is hard for the level designers to create content on time and without errors. This is because Unity, or the available third-party tool you use, doesn't satisfy all the required functionalities.

Sometimes, because you have more people working on the project, the lack of a mechanism to encourage people to follow standards makes your video game crash constantly.

In the same scenario, imagine that your project also requires a lot of art assets, so artists constantly add these to Unity. The problem appears later when one of the developers needs to constantly check whether the settings of these assets are configured properly to make these look right in the final build, consuming development time.

Finally, your project will be available on several platforms. However, owing to the specific characteristics of your video game, every time you make a production build, you must check whether all the settings are okay. You also need to check whether you removed all the cheat menus used by your testers and that the correct assets are loaded into each because you are preparing a trial version. Managing this becomes a huge task!

To solve all these issues, Unity provides an editor scripting API. Using this we can do the following tasks:

- Modify how the Unity editor behaves, triggering our code with specific events
- Improve the workflow assistance with a custom GUI that seamlessly integrates with the Unity editor GUI
- Automate repetitive tasks by accessing the Unity editor's main functionalities

Understanding how to use the editor scripting API to create editor scripts in your project will allow you to make Unity work for your video game and boost the productivity of the video game development.

Editor scripting basics

It's time to go hands on in the creation of editor scripts so in this section we are going to explore how to start them off.

What is an editor script?

An **editor script** is any piece of code that uses methods from the **UnityEditor** namespace, and its principal objective is to create or modify functionalities in the Unity editor.

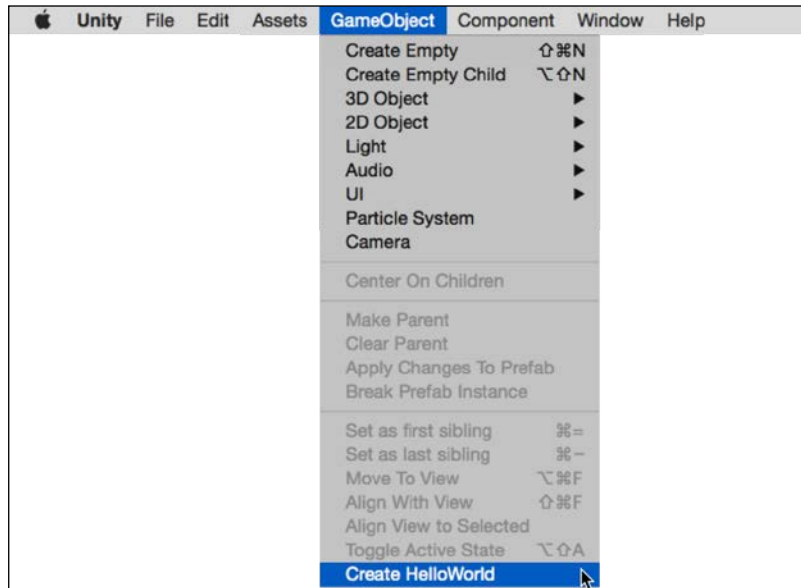
To see this working, let's start with a basic example. Create a new project in Unity and then a new script called `HelloWorld.cs`. Don't worry about where to place the script, we'll talk about that in a bit. Copy the following code:

```
using UnityEngine;
using UnityEditor;

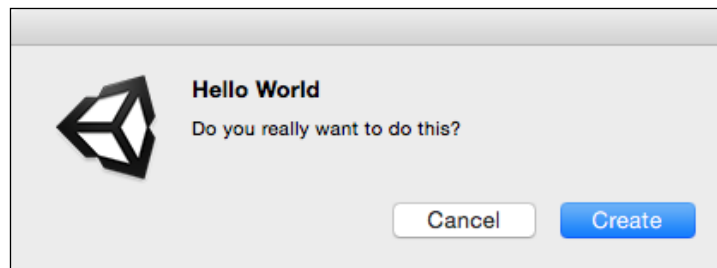
public class HelloWorld {

    [MenuItem ("GameObject/Create HelloWorld")]
    private static void CreateHelloWorldGameObject () {
        if(EditorUtility.DisplayDialog(
            "Hello World",
            "Do you really want to do this?",
            "Create",
            "Cancel")) {
            new GameObject("HelloWorld");
        }
    }
}
```

Wait for the compiler to finish and then go to the Unity editor menu and click on **GameObject**. At the end of the menu, you will see an item called **Create HelloWorld**, as shown in the following screenshot:



Click on this item, then a dialog window asks whether you really want to create this game object:



After clicking on **Create**, a new game object with the name **HelloWorld** is added to the current scene. You can check this in the **Hierarchy** window: