Moving from Unity to Godot

An In-Depth Handbook to Godot for Unity Users

Alan Thorn

Apress®

Moving from Unity to Godot

An In-Depth Handbook to Godot for Unity Users

Alan Thorn

Apress®

Moving from Unity to Godot: An In-Depth Handbook to Godot for Unity Users

Alan Thorn
High Wycombe, UK

ISBN-13 (pbk): 978-1-4842-5907-8 ISBN-13 (electronic): 978-1-4842-5908-5
https://doi.org/10.1007/978-1-4842-5908-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Spandana Chatterjee
Development Editor: James Markham
Coordinating Editor: Divya Modi

Cover designed by eStudioCalamar

Cover image designed by Pixabay

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-5907-8. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

Table of Contents

About the Author

Alan Thorn is an expert on leading technical teams for game development. He previously worked at Microsoft, Teesside University, Apress Publishing, and Disney. Alan specializes in helping “tech heads” thrive and flourish in their chosen fields. With 18-year game industry experience, Alan has written 28 books, presented 30 online courses, and created 33 games including the award-winning adventure *Baron Wittard: Nemesis of Ragnarok*. Alan is dedicated to helping creative people make high-impact experiences. He was a Studio Director at Wax Lyrical Games and a Senior Author at LinkedIn Learning, and now he leads the prestigious MA program for Games Design and Development at the BAFTA-winning National Film and Television School, an incubation space for breakthrough gaming talent. Alan is a pioneer of the new “Open Stream” model of Transformative Learning, and he advises in higher education on disruptive curriculum content and instructional design. Alan speaks passionately worldwide about the future of interactive experiences. In this book, he clearly details Godot-specific terminology, how to use its interface effectively, how scenes are structured, coding in C#, and optimal ways of working.

About the Technical Reviewer

Doug Holland is a Software Architect at Microsoft Corporation in the One Commercial Partner team. In his role at Microsoft, Doug provides guidance to Microsoft's partners looking to digitally transform with cloud computing, mixed reality, and other emerging technologies. Doug holds a master's degree from Oxford University in Software Engineering and lives in Northern California with his wife and children.

Introduction

Congratulations on joining me to take an exciting journey, one that moves from Unity to the Godot engine. Godot is a completely free, open source, and cross-platform engine that's making amazing strides and progress. Godot can make 2D and 3D games, and it can be used for other products like visualizations, Movie Previz, historical recreations, and more. In the competitive climate of game development, where new tools and technologies are continually reshaping the landscape, and challenging established norms, it can be difficult knowing which tools to learn or to trust in. Every software, it seems, has its rise and fall. It's difficult knowing which one represents the actual future and who is just a passing fad. But your decisions about which software to use are crucial for your business and your success. Your choice of engine influences *your* future. It influences how quickly and easily you can work, which platforms your end product can target, and which tools you can integrate with.

One reason so many people today are switching from Unity to Godot, or at least considering the switch, is because Godot represents a vision and a promise, a free engine that's open source and community driven, an engine that's easy to use and reliable, and an engine that's open about its development road map.

One of the historic fears surrounding the use of any game engine – as opposed to making your own engine – was its closed and proprietary nature, that it would keep you locked into specific ecosystems and to specific agendas led by companies and parties outside of your own business. And until recently, you never really had much choice if you wanted to make a professional-grade game. You had to lock yourself to a closed engine driven by people behind closed doors. But Godot changes that, and that's why it's an especially important engine today. It represents an opportunity to make games on very different terms. And with the outstanding success of other open source tools, like Blender, we have every reason to feel excited for the future of Godot. I'm truly glad you're going to join me on this journey.

CHAPTER 1

Introducing Godot: Why Migrate?

This book helps you convert easily from Unity to Godot. It assumes you're already familiar with Unity (at least the basics) but are completely new to Godot. This book translates Unity terminology to Godot terminology. Plus, it features comprehensive tutorials and guides to get you started quickly in Godot. Godot is a completely free and open source game engine that's growing rapidly in popularity, acceptance, and adoption, especially for independent games. More and more developers worldwide are happily joining the Godot community to make great games together, and there's never been a better time to start learning. In this chapter, we explore what Godot is, how to download and install it, and strong reasons why we should use it compared to other engines, like Unity or Unreal. So let's get started.

This book was written for the Godot 3 release cycle. This includes 3.0, 3.1, and later releases in the 3 cycle. You may be using a later release than the one featured in this book; but most of the content presented here should be good for the near future.

Getting and Installing Godot

If you're using Unity, you probably already know what Godot is! Godot is a game engine. It's software for making games and interactive experiences. It features a world creator, a graphical interface, a code editor, a complete API, and build tools for deploying your game to different platforms as a stand-alone application. In short, imagine a game engine that's very similar to Unity in its ease of use, but it's completely free and open source. That's Godot.

You can download Godot from its home page https://godotengine.org/. Simply navigate a browser there and click the *Download* button from the top menu. This takes you to the Download page. From there, be sure to download the *Godot Mono Version C#*. See Figure 1-1.

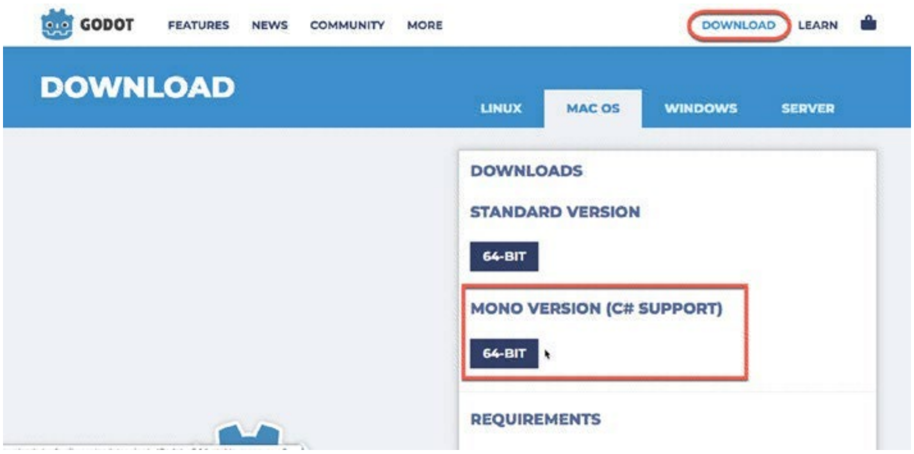


Figure 1-1. *Download and Install the C# Godot Version for Your Operating System*

Multiple versions of Godot are available for download, all for multiple operating systems. Specifically, Godot natively supports Windows, Mac, and Linux; and it's available in a *Standard Version* and a *Mono Version*. The Mono Version lets you make script files in the C# scripting language – like Unity – while the Standard Version supports Godot's custom language, GDScript, only. This book uses the Mono Version and writes script files in C#. Using C# not only makes the transition from Unity to Godot simpler; it's also a great language offering premium level runtime performance. So I recommend choosing the Mono Version in all cases. Godot features full support for C# 8.

Why Use Godot?

So why use Godot at all? That's a good question! After all, there're tons of apparently free and spectacular game engines available today. For example, you can download *Unity*, *Unreal*, or *Lumberyard* right now and make games with them. These engines are well documented, widely used, and are responsible for many well-loved games on the market. So why use Godot instead? This section lists important reasons for choosing Godot as opposed to its alternatives.

Godot Is Free

Godot is completely free of cost. You pay nothing for using Godot or for selling your Godot games. Really. This is actually very different from Unity and Unreal even though many people think they're fully free. Unity and Unreal are, in fact, not completely free. True, you may download and use them *free of charge*, but you'll need to pay up if you release a game commercially and its revenue surpasses a specific threshold. The amount differs per engine. But let's put this into perspective as of November 2019. For Unity, if your company earns more than *100,000USD* in a single financial year, then you'll need to purchase a Unity Professional License. This costs *125USD* per month. See the Unity FAQ page here: `https://unity3d.com/unity/faq/2491`. In future, this pricing structure could increase! Similarly, for Unreal Games, you'll need to pay 5% of your gross revenue (before tax) after you earn 3,000USD (see www.unrealengine.com/en-US/faq). This could amass to a huge sum if your game becomes successful! So, neither Unity nor Unreal is fully free. But Godot is. You pay nothing for using Godot, ever. Simple.

Godot Is Open Source

Godot is completely open source. This means you can download, inspect, edit, and compile its source code immediately. Plus, you can use and redistribute any derived versions you may make. Open source should be reassuring to a game developer. This is because their work won't depend on a closed foundation of proprietary code that's maintained by a third-party developer whose agenda may be very different to yours. The engine is licensed under the MIT License, which is summarized by Godot:

> *You are free to download and use Godot for any purpose, personal, non-profit, commercial, or otherwise. You are free to modify, distribute, redistribute, and remix Godot to your heart's content, for any reason, both non-commercially and commercially.*
>
> ***—Godot website (https://docs.godotengine.org/en/3.1/about/faq.html)***

This book doesn't cover compiling Godot from source. It assumes you'll be using a fully built version that's downloadable from the Godot website. For more information on compiling from source, visit the Development website here: https://docs.godotengine.org/en/latest/development/compiling/getting_source.html.

Godot Is Evolving

Godot is always evolving. It was established in 2014 by software developers Juan Linietsky and Ariel Manzur, and it's continued to grow in popularity and features year after year. Godot has a full range of features that any Unity developer will recognize and expect from an engine, plus really interesting features that may pleasantly surprise you. Godot supports Light Baking, full Global Illumination, Navigation Meshes and Path Finding, Visual Scripting, Constructive Solid Geometry (CSG), and 2D functionality for sprites and User Interfaces. Godot currently supports two scripting languages, namely, GDScript and C#; the former is more established in the engine, while the latter is newer. Godot is actively maintained by a strong development community, and many people financially support the project through *Patreon*, here: www.patreon.com/godotengine.

Godot Is Supported

Godot is strongly supported by a growing community of game developers, educators, and creative evangelists. Together they have a ton of experience. They have a shared purpose in promoting Godot, helping each other, and helping newcomers learn the tools. Online documentation, books like this, videos, and video courses are all available to help you learn Godot from scratch and to continue using Godot professionally. Godot has comprehensive online documentation that should be used frequently as a reference, in conjunction with this book, as you progress chapter by chapter. It's available here: https://docs.godotengine.org. Furthermore, you'll also want to check out https://godotsharp.net for a C#-oriented reference and also my YouTube channel BeIndie.biz (www.youtube.com/channel/UCF1X3sTIj-pCIcR_C2wg6SQ/) for regular tutorials on Godot.

Godot Is About Games and Experiences

If you attend game conferences, you'll find a growing community of game developers discontented with the strategic direction of both Unity and Unreal right now. These developers are seeking alternative options, and understandably so. There're several important reasons for this search worth mentioning here. Some consider Unreal too expensive and technically burdensome, especially for small development teams with less specialist needs. Some see Unity as focusing too heavily on architectural and automotive visualization features at the expense of games. And some feel that both engines have just gotten too wrapped up in the arms race toward photorealism, the sense that both are "growing too big to fully understand the needs of independent developers and small teams." There's a sense, either real or perceived, that in both Unreal's and Unity's most recent growth (especially in the latter), there's also a significant detachment from their original user base. In this search, then, many people are now finding their ideal in the Godot engine – driven by and built by the game development community. Godot is first and foremost a *game engine*. Its structure, tools, feature set, and build options have games (and interactive experiences) in mind from the outset. Its source code is open and its development road map are constantly under an open discussion by the developer community. The full development road map can be found on GitHub here: `https://github.com/godotengine/godot-roadmap`.

Godot and C#

As a Unity developer, you'll be familiar with *C#* for scripting – unless you use a Visual Scripting plug-in like *Playmaker*. C# is a versatile, powerful, and easy-to-learn language that delivers comparatively excellent runtime performance. And so it's likely you'll want to continue using it on migrating to Godot. After all, you probably don't want to learn a completely new language and syntax just because you're moving to a new engine. Thankfully, you can bring your C# knowledge with you! This is because the Godot 3.1 Mono Build (or higher) supports C# as a scripting language. However, there are some important limitations to using C# that you should know. These limitations may be ironed out in later releases, but as of 3.1 they apply:

1. You can't build C# projects for the Web or mobile devices.

   Godot lets you build games for many platforms, including desktop, Web, and mobile – using the GDScript language. However, C# projects build only for desktop systems – both Web and mobile are currently excluded. C# support may be added soon, but this is in development.

You can keep track of the development status for these features, as follows:

Android

https://github.com/godotengine/godot/issues/20267

iOS

https://github.com/godotengine/godot/issues/20268

HTML5

https://github.com/godotengine/godot/issues/20270

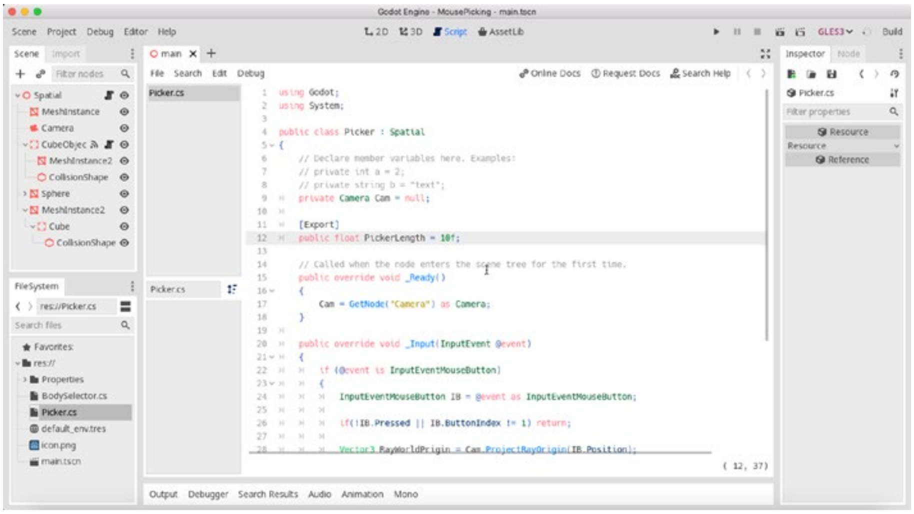
---

2. You won't get C# code completion or syntax highlighting for the native script editor.

   Unlike Unity, Godot provides an integrated text editor for creating scripts as part of the main editor interface; see Figure 1-2. This is the script editor window. You can use this window to type C# script files, but it lacks full language support. It's designed for GDScript. So you won't get full syntax highlighting or code completion or any coding assistance for C#. To get this, you'll need to use external code editor! This book will show you how to use **Visual Studio Code** with Godot – complete with syntax highlighting and code completion. Visual Studio Code is supported on Windows, Linux, and macOS.

3. It's easy to break the metadata for your C# project.

   As you add and remove files to and from your project, Godot maintains metadata. This tracks connections between files and resources, and it maintains lists of script files needed for compiling. In C# projects, it's currently easy for anybody to accidentally break the metadata. This causes compilation to fail. Don't worry though. These issues can usually be fixed easily and manually, and we'll see how soon.

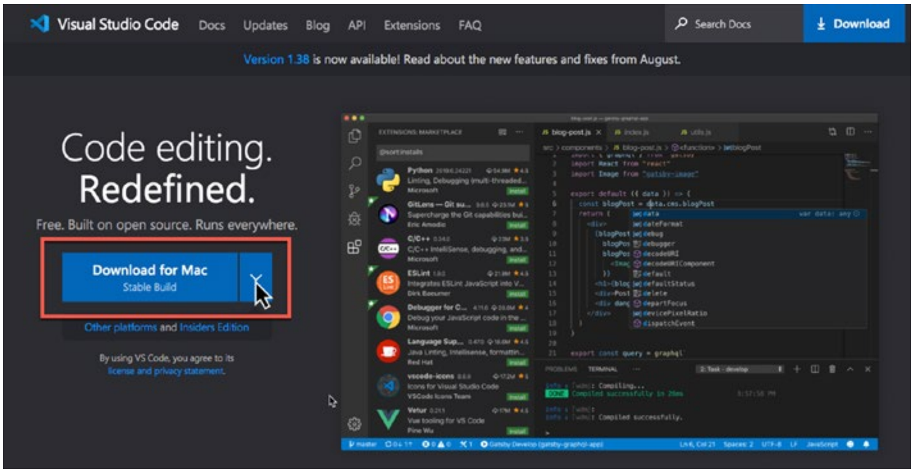


```
using Godot;
using System;

public class Picker : Spatial
{
    // Declare member variables here. Examples:
    // private int a = 2;
    // private string b = "text";
    private Camera Cam = null;

    [Export]
    public float PickerLength = 10f;

    // Called when the node enters the scene tree for the first time.
    public override void _Ready()
    {
        Cam = GetNode("Camera") as Camera;
    }

    public override void _Input(InputEvent @event)
    {
        if (@event is InputEventMouseButton)
        {
            InputEventMouseButton IB = @event as InputEventMouseButton;

            if(!IB.Pressed || IB.ButtonIndex != 1) return;

            Vector3 RayWorldPrigin = Cam.ProjectRayOrigin(IB.Position);
```

Figure 1-2. *The Godot Editor Features a Scripting Window for GDScript and C#. However, Third-Party Editors Are Better for C#, As We'll See*

Getting Started with C# in Godot

So you've finally downloaded and installed the Godot Mono Version for your OS, as mentioned earlier in this chapter. Great! Let's now take the next steps to configure Godot for C# scripting. Once set up, we'll be ready to start making games. Open up a web browser and download Visual Studio Code to your computer. See Figure 1-3. The URL is `https://code.visualstudio.com/`.



Version 1.38 is now available! Read about the new features and fixes from August.

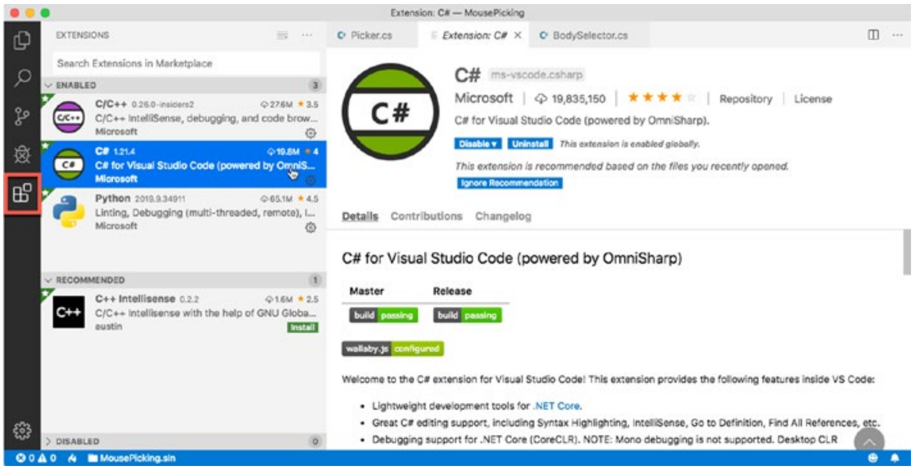
Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

By using VS Code, you agree to its license and privacy statement.

Figure 1-3. *Downloading Visual Studio Code for Your Operating System. This Book Supports All Platforms for Godot, Although I'm Using a Mac for Screenshots and Examples*

After running Visual Studio Code for the first time, select the Extensions tab and install the *OmniSharp C# Extension* for syntax highlighting and code completion support. See Figure 1-4.



C# for Visual Studio Code (powered by OmniSharp).

This extension is enabled globally.

This extension is recommended based on the files you recently opened.

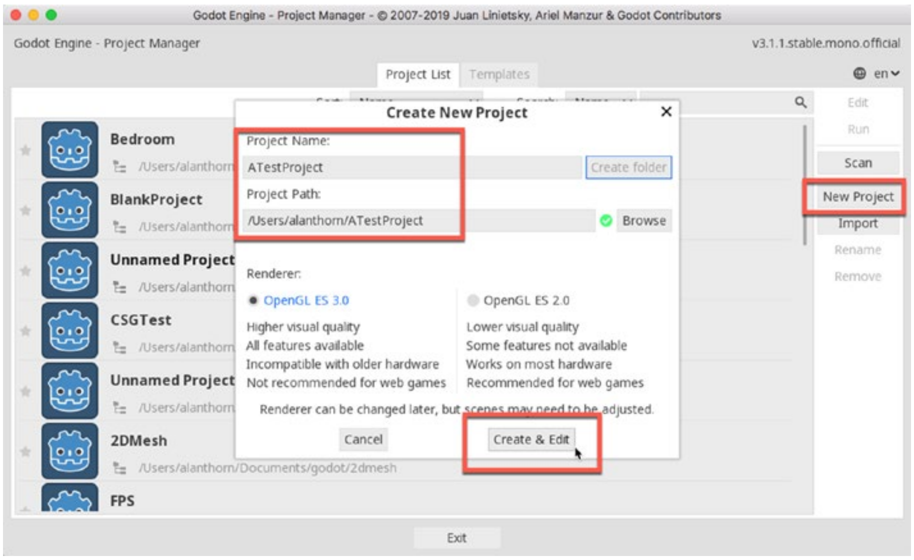
C# for Visual Studio Code (powered by OmniSharp)

Welcome to the C# extension for Visual Studio Code! This extension provides the following features inside VS Code:

- Lightweight development tools for .NET Core.
- Great C# editing support, including Syntax Highlighting, IntelliSense, Go to Definition, Find All References, etc.
- Debugging support for .NET Core (CoreCLR). NOTE: Mono debugging is not supported. Desktop CLR

Figure 1-4. *Installing the C# OmniSharp Extension for Visual Studio Code*

Next, Open Godot and create an empty project from the Project Creation Window. Click the **New Project** button and enter your project name and location. You can name your project anything and store it anywhere. We only need to create a new project to access the Editor Settings window from the main interface. See Figure 1-5.


Godot Engine - Project Manager - © 2007-2019 Juan Linietsky, Ariel Manzur & Godot Contributors
Godot Engine - Project Manager
v3.1.1.stable.mono.official
Project List
Templates
en
Edit
Run
Scan
New Project
Import
Rename
Remove
Bedroom
BlankProject
Unnamed Project
CSGTest
Unnamed Project
2DMesh
/Users/alanthorn/Documents/godot/2dmesh
FPS
Create New Project
Create folder
Browse
Cancel
Create & Edit
Exit

Project Name:
ATestProject
Project Path:
/Users/alanthorn/ATestProject

Renderer:
● OpenGL ES 3.0 | ○ OpenGL ES 2.0
Higher visual quality | Lower visual quality
All features available | Some features not available
Incompatible with older hardware | Works on most hardware
Not recommended for web games | Recommended for web games

Renderer can be changed later, but scenes may need to be adjusted.

Figure 1-5. *Creating a New Godot Project*

From the main Interface, select the menu item *Editor* ➤ *Editor Settings*. This displays the Editor dialog where you can customize application settings. See Figure 1-6.

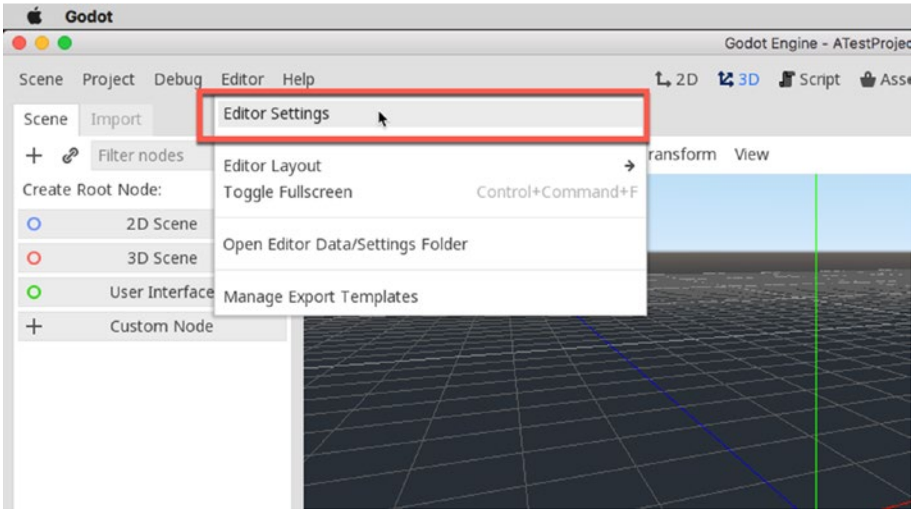


Figure 1-6. *Accessing the Editor Settings Window*

From the **Editor Settings** Window, choose *Mono* ➤ *Editor* from the list view and then select Visual Studio Code from the **External Editor** drop-down menu. This sets Visual Studio Code as the default code editor for Godot. This means that Visual Studio Code will open automatically whenever you open script files from the Godot **File System** (Project Panel). See Figure 1-7.

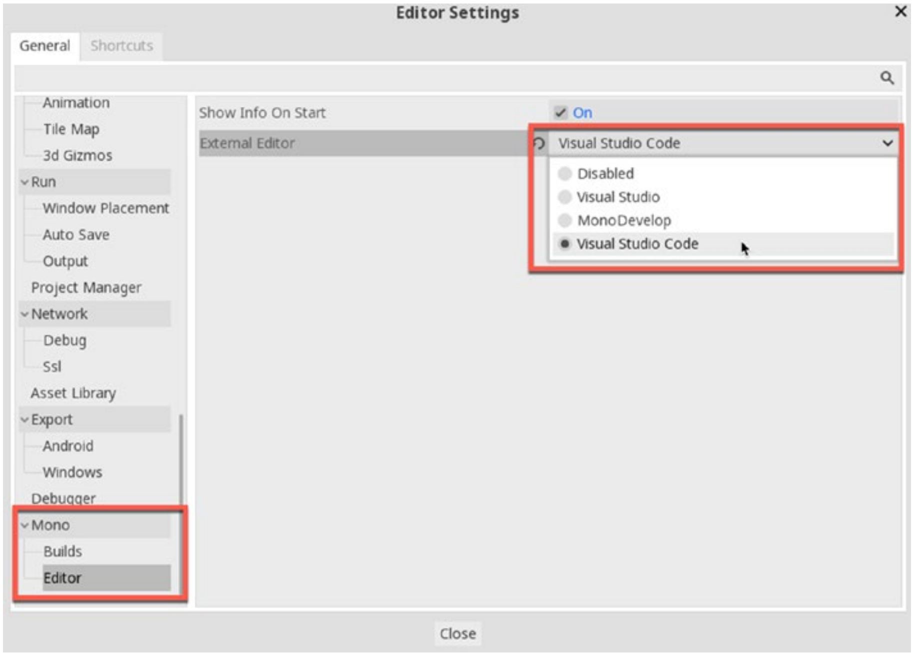


Figure 1-7. *Setting the Default Code Editor*

Excellent! You've now configured Visual Studio Code as the default code editor for Godot. This editor will now open to display script files in your project. You're now all set to get started!

You can view a free YouTube video tutorial on configuring Visual Studio Code on my BeIndie.Biz channel here: `www.youtube.com/watch?v=ra-BJ-fJ6Qo&t`.

Summary

On reaching this point, you've now downloaded and installed Godot Mono, and you're ready to start coding games using C# for any desktop platform – Windows, Linux, or Mac. Godot is an excellent, free, and open source engine; and by using this tool, you can create great products.

Now let's jump in and learn Godot quickly. I'll be converting your Unity knowledge to Godot knowledge. Let's go!

CHAPTER 2

Godot Fundamentals

So you've now installed Godot and Visual Studio Code. That's great! Chapter 1 explained how to configure Godot on your computer – Windows, Mac, or Linux – and how to create Godot projects successfully. This chapter picks up from the previous to get you started quickly. Specifically, it's a high-powered conversion course. It helps you move from the already familiar world of Unity into the new world Godot. In this chapter, you'll learn how to create Godot projects, how to use the editor interface, how to convert from Unity's language and terminology into Godot's, and how to build scenes with basic game functionality. If you've never used Godot before, then you absolutely need to read this chapter. So let's go.

Note Even if you've used Godot before, it's strongly recommended that you read this chapter. It covers Godot's most core features, comparing them to Unity's and also making important distinctions that you need to know about.

Unity comes with a whole bunch of names and words you'll already know, such as *GameObject, Component, Scene, Script, Asset, Hierarchy, Inspector*, and more. Godot unsurprisingly has exactly the same concepts and ideas! But it sometimes uses different names for them. Let's start then by exploring the names we'll be encountering extensively throughout this chapter. Table 2-1 lists how Unity's names and terms compare to Godot's. Later, we'll see more in-depth explanations for them and also make important distinctions.

Table 2-1. *Terminology: Unity vs. Godot*

Unity	Godot
Project	Project
Scene	Scene
GameObject	Node
Prefab	Scene
Hierarchy	Scene Tree
Project Panel	File System
Inspector	Inspector
Empty Object	Spatial Node
Asset	Resource
Tag	Group

Godot Projects

The first step in working with Godot is to create a new **Project**, just like working with Unity. In Godot, "One Project" refers to "One Game" or "One Simulation," or – more generally – *one complete experience*. By creating a new Project, Godot effectively creates a folder in your computer's file system, and it will contain all necessary files for your game, including metadata, textures, meshes, sounds, animations, and more. If you ever want to share your project with other team members or friends – allowing them to open it inside Godot for editing – then you'll need to send them your complete project folder. To create a new Project, simply launch the Godot application, and you'll be presented with the Project Creation

Introduction Window. This is basically Godot's version of the **Unity Hub**. From this Window, select the **Projects** tab and click the **New Project** button from the right-hand margin. See Figure 2-1.

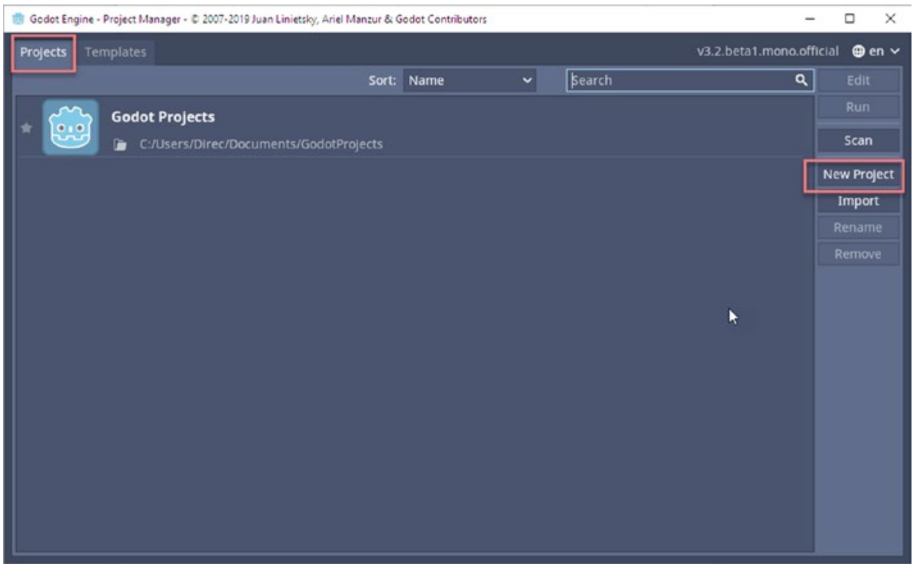
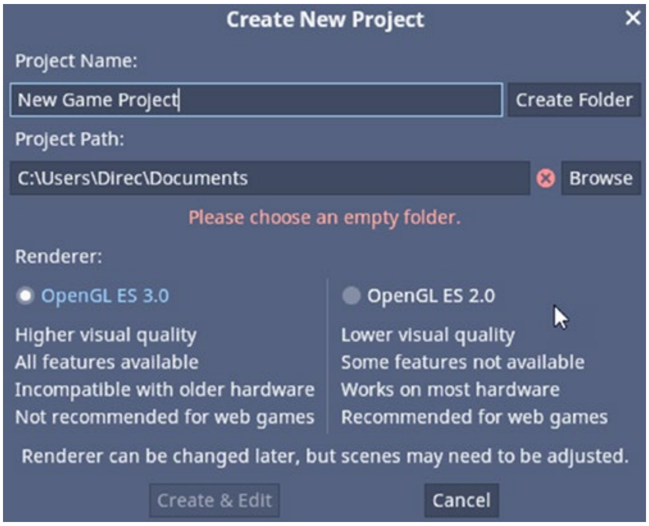

Godot Engine - Project Manager - © 2007-2019 Juan Linietsky, Ariel Manzur & Godot Contributors

Figure 2-1. *Creating a New Godot Project*

The project creation dialog is small and contains very few options. But they're important. By default, the confirmatory **Create & Edit** button in the bottom-left corner will probably be disabled, requiring you to make some changes. See Figure 2-2.


Create New Project

Project Name: New Game Project

Project Path: C:\Users\Direc\Documents

Please choose an empty folder.

Renderer:

● OpenGL ES 3.0
Higher visual quality
All features available
Incompatible with older hardware
Not recommended for web games

○ OpenGL ES 2.0
Lower visual quality
Some features not available
Works on most hardware
Recommended for web games

Renderer can be changed later, but scenes may need to be adjusted.

Figure 2-2. *The Godot Project Creation Window*

Specify a project name using the **Project Name** text field and then click the **Create Folder** button next to the field if the **Project Path** location is suitable for you. The Project Path specifies the folder on your computer's file system where a new sub-folder will be created, matching the Project Name. It will contain all project files and become the Project Folder. Be sure to leave the option **OpenGL ES 3.0** enabled, unless you're making a game for very old hardware, older mobile hardware, or web games. In this book, we won't be doing that! Godot supports Windows, Mac, Linux, Android, Web, and (soon) iOS. See Figure 2-3.