Unreal Engine 4 Game Development Essentials

Unreal Engine 4 is a complete suite of game development tools that give you the power to develop your game and seamlessly deploy it to iOS and Android devices. It can be used for the development of simple 2D games or even stunning high-end visuals. Unreal Engine features a high degree of portability and is therefore in demand.

This book will introduce you to the most popular game development tool called Unreal Engine 4 with hands-on instructions for building stunning video games. You will begin by creating a new project or prototype by learning the essentials of Unreal Engine by getting familiar with the UI and Content Browser. Next, we’ll import a sample asset from Autodesk 3ds max and learn more about Material editor. After that we will learn more about Post Process. From there we will continue to learn more about Blueprints, Lights, UMG, C++ and more.

What you will learn from this book

- Download both the binary and source version of Unreal Engine 4 and get familiar with the UI
- Get to know more about the Material editor and how it works
- Add a post process to the scene and alter it to get a unique look for your scene
- Acquaint yourself with the unique and exclusive feature of Unreal Engine 4—Blueprints
- Find out more about static and dynamic lighting and the difference between various lights
- Use Matines to create cut scenes
- Create a health bar for the player with the use of Unreal Motion Graphics
- Get familiar with Cascade Particle Editor

Who this book is written for

If you have a basic understanding of working on a 3D environment and you are interested in video game development, then this book is for you. A solid knowledge of C++ will come in handy.


Satheesh PV
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introduction to Unreal Engine 4'
- A synopsis of the book’s content
- More information on Unreal Engine 4 Game Development Essentials
About the Author

**Satheesh PV** is a game programmer living in Mumbai, India. He was selected by Epic Games as one of the closed beta testers for Unreal Engine 4 before its public release. He started his career as a game developer in 2012 by making a first person multiplayer game with his brother and close friend using Unreal Development Kit. He also created Unreal X-Editor, which was an IDE developed for UnrealScript, the native scripting language of Unreal Engine 3. He is also a moderator at Unreal Engine forums as well as a spotlight member and engine contributor.
The purpose of *Unreal Engine 4 Game Development Essentials* is to teach people interested in using Unreal Engine how to create video games. You will learn what Unreal Engine is and how to download and use it. From there, we will go through the collection of tools available in Unreal Engine 4 including Materials, Blueprints, Matinee, UMG, C++, and more.

**What this book covers**

*Chapter 1, Introduction to Unreal Engine 4*, is where we begin our journey on *Unreal Engine 4 Game Development Essentials*. In this chapter, the reader will learn how and where to download Unreal Engine as well as the difference between the source version and launcher version. After the Engine's installation (or compilation, if it was the source version) we will get comfortable with the user interface of Unreal Engine. We will also learn about the basics of Content Browser, BSP, and how to change the splash screen and the icons for your game.

*Chapter 2, Importing Assets*, teaches how to import your custom FBX assets into Unreal Engine once we get the Engine up and running. You will learn about collisions, materials, and the level of detail.

*Chapter 3, Materials*, teaches you about the Material editor and some common nodes used to create shaders for your assets. After learning the basics of Material, we will create an example material function that can change the intensity of a normal map.

*Chapter 4, Post Process*, continues to post-processing after teaching you about materials. In this chapter, you will learn how to override the default post process settings. After that, we will learn how to add our own post process volume and learn a simple but very powerful feature called LUT. After that, we will create a special material that can be used with post process, and this material will have the ability to highlight user-defined objects in the world.
Chapter 5, *Lights*, gets us halfway through our *Unreal Engine 4 Game Development Essentials* journey, and this chapter will introduce you to the lighting system. We start off by covering the basics, such as placing lights and going through the common settings. You will then learn more about the Lightmass Global Illumination system, including how to properly prepare a UV channel for your asset to be used with Lightmass. By the end of this chapter, you will learn how to build your scene with Lightmass as well as Lightmass settings.

Chapter 6, *Blueprints*, teaches you what Blueprints are and about the various types of Blueprints that are available in the Engine. Blueprints are Unreal Engine’s number one tool that allows artists and designers to quickly prototype their game (or even make one!). You will also learn about the different graph types, such as event graph, function graph, macro graph, and so on, and how to spawn a Blueprint dynamically at runtime.

Chapter 7, *Matinee*, looks at the cinematic side of Unreal Engine 4 and the tool associated with it, called Matinee. You will learn what Matinee is, how to create one, and get familiar with the UI. After the basics, we will learn how to manipulate objects in Matinee as well as create a very basic cutscene, which we will trigger using Blueprints.

Chapter 8, *Unreal Motion Graphics*, teaches you to create a basic HUD that shows the health of the player. Unreal Motion Graphics (UMG) is the UI authoring tool in Unreal Engine. UMG is used to create Player HUD, Main Menu, Pause Menu, and so on. You will also learn how to create 3D widgets, which can be placed in the world or attached to an actor class.

Chapter 9, *Particles*, looks at the extremely powerful and robust tool called cascade particle editor and creates a particle system, as no game is good without good visual effects. We then combine this with simple Blueprint scripting to create randomly bursting particles.

Chapter 10, *Introduction to Unreal C++*, goes over C++ as we draw close to the end of our *Unreal Engine 4 Game Development Essentials* journey. In this chapter, you will learn how to get Visual Studio 2015 Community Edition and learn the basics of C++ by inspecting the Third Person Template character class. We will then extend this class to add support for health and the health regeneration system. You will also learn how to expose variables and functions to Blueprint Editor.

Chapter 11, *Packaging Project*, brings us to the end of our *Unreal Engine 4 Game Development Essentials* journey. In this final chapter, we will recap all the things we’ve done, including a few tips, and finally, you will learn how to create a release version of your game.
Welcome to Unreal Engine 4 Game Development Essentials. In this chapter, you will learn how to download Unreal Engine's source version and launcher version. After that, we will get familiar with the Unreal Engine 4 UI and Content Browser.

Unreal Engine 4 download

Unreal Engine 4 is completely free (including all future updates!) to download and use. You get all the Unreal Engine tools, free sample contents, complete C++ source code which includes code for the entire editor, and all of its tools; you also get access to official documentation that includes tutorials and support resources, plus you get access to UE4 marketplace, which offers tons of free and commercial content.

Unreal Engine 4 can be downloaded in two different versions. One is a binary version (launcher) and the other is the source version (GitHub). The differences between the GitHub and launcher version are as follows:

- **Launcher (binary) version:** These are compiled by Epic and are available through launcher. You will also get all source files (*.cpp) with the launcher version, but you cannot make any modifications to Unreal Engine since launcher versions do not generate a solution file.

- **GitHub version:** These do not have any binary files so you have to compile the Engine yourself. You get the entire source and you can modify virtually anything in Unreal Engine. You can add new Engine features, modify existing features or remove them (which no one does), and create a pull request on GitHub so if Epic likes it, they will integrate it officially into Unreal Engine.

In this guide, I'll show you how to get both versions.
Introduction to Unreal Engine 4

**Downloading the launcher version**

To download the launcher version of Unreal Engine, you obviously need the launcher. To download the launcher, follow these steps:

1. First go to https://www.unrealengine.com/ and log in using your credentials.
2. Once you log in, you can download the launcher by clicking on the big orange **Download** button under **Get Unreal Engine**.

When you open the launcher for the first time after installation, it should automatically download the latest version of Unreal Engine 4. If it doesn't, then go to the **Library** tab and click on **Add Engine**. A new Engine slot will now appear and here, you can select your Unreal Engine version and install it.
# Downloading the GitHub version

To download the source of Unreal Engine 4, follow these steps

1. First create a GitHub account (it's free!).
2. After that, you need to go to https://www.unrealengine.com/dashboard/settings and update your GitHub account name and click on **Save**:

<table>
<thead>
<tr>
<th>Download</th>
<th>Profile</th>
<th>Billing</th>
<th>Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Forking Unreal Engine repository

After you have successfully linked your GitHub account with your Unreal Engine account, you need to log in to GitHub and navigate to the Unreal Engine repository.

Make sure you have linked your GitHub account to your Unreal Engine account. Otherwise, you will not be able to see the Unreal Engine repository.
Introduction to Unreal Engine 4

When you are at the repository page:

1. Click on **Fork** at the top right of the page.
2. Select your username to fork it to your GitHub repository.
3. Then, you need to download GitHub for Windows (if you are on Windows) or GitHub for Mac (if you are on Mac) and install it.

You need this Git client to clone (download) your forked repository, make your own changes to Unreal Engine, and submit the changes as a pull request to Epic to integrate them into the editor.

To clone your forked repository follow these steps:

1. Start GitHub and log in using your credentials.
2. Click on the plus (+) sign on the top left corner of the Git client.
3. Then, click on the **Clone** tab and select your **username** (you should now see Unreal Engine).
4. Now, click on **Clone** Unreal Engine and choose a folder where you want to save your Unreal Engine repository.
5. Click on **OK**.
6. You should now see GitHub cloning Unreal Engine to your hard disk.
Once cloning is complete, navigate to that directory and run the Setup.bat file.

1. This will download all the necessary files that are needed to compile the engine and will also install all the required prerequisites for the Engine.
2. This might take some time depending on your Internet speed because it has to download more than 2 GB of files.

**Compiling Unreal Engine**

Once Setup.bat has finished, run GenerateProjectFiles.bat, which will generate the Visual Studio Solution file. Open the UE4.sln file and now, you are all set to compile your own copy of Unreal Engine 4 ☺. Now, right-click on UE4 in the Solution Explorer and click on Build.

This will take from 15 minutes to 1 hour depending on your system hardware. So sit back, grab a cup of coffee, and wait till Engine finishes compiling.

**Getting familiar with Unreal Engine**

Once your Engine finishes compiling (or downloading, if you are using launcher) it's time to start it:

- **Starting your custom build**: You can either press F5 in Visual Studio to start debugging the Engine or navigate to the directory where you downloaded it and go to Engine\Binaries\Win64 folder and double-click on UE4Editor.exe.
- **Starting launcher build**: Simply click on that big Launch button and you're good to go.
You might experience long loading time when you start the Engine for the first time after compiling. This is because Unreal Engine will optimize the contents for your platform to derive data cache. This is a one-time process.

After the splash screen, you should now see the Unreal project browser. Perform the following steps:

1. Select the **New Project** tab, and this is where you create your new projects.
2. For this book, we will stick with a **Blank Blueprint Project**. So, in the **Blueprint** tab, select **Blank** project.
3. You can choose which platform you want for your project. There are two platforms available: **Desktop/Console** and **Mobile/Tablet**. Feel free to change this setting for your project. The second setting determines the graphics settings for your platform. If you choose **Desktop/Console**, it's better to stick with **Maximum Quality** and if your project is targeting **Mobile/Tablets**, you should choose scalable 3D or 2D, which is aimed at low-end GPUs. The third and final setting lets you add some **Starter Content** from Epic, which contains some basic meshes, materials, and textures. You can choose not to include **Starter Content** so the project will only contain essential elements for the selected project.
4. Note that it is not recommended to include **Starter Content** when creating a project for the **Mobile/Tablet** platform. This can significantly increase the package size of your project.
5. Choose a name for your project and the location where you want to save it.
6. Finally, click on **Create Project** to start Unreal Engine 4 with your project:

This is how the Unreal Engine user interface looks:
Once Unreal Engine starts up, you should see a scene similar to the preceding screenshot. This is the scene that will be displayed by default, if you choose to include Starter Content. If you skip Starter Content, then the startup scene will be different.

The viewport toolbar
The viewport toolbar contains various tools that you will use throughout your level design process. Let's take a quick look at them:

- **Transform Tools**: These three tools are the move tool, the rotate tool, and the scale tool.
- **Coordinate System**: This allows you to move, rotate, or scale your Actor either on world axes (world space) or on its own local axes (local space). By default, Unreal editor starts in world axes but you can toggle by clicking on the icon. The globe icon means world space and the cube icon means local space.
- **Snapping and Move Grid**: Snapping allows you to snap one Actor to another Actor’s surface, and move grid allows you to snap to a three-dimensional implicit grid within the scene.
- **Rotation Grid**: This provides incremental rotation snaps.
- **Scale Grid**: This snaps to additive increments.

Snapping preferences for move, rotate, and scale can be adjusted in Editor Preferences. Go to Edit | Editor Preferences | Viewports and then scroll to Grid Snapping Category.

- **Camera Speed**: This lets you control how fast the camera moves in viewport.

You can fine-tune the camera speed by holding down the right mouse button (while using WASD controls) and scrolling the mouse wheel up or down to speed up or slow down the camera’s movement.
• **Maximize Viewport:** This toggles between a single viewport and a 4-view split style.

You can adjust the layout of **Viewport** by changing the **Layout** option, as shown in the following screenshot:

Later in this chapter, you will learn how to use **Binary Space Partitioning (BSP)** and change some project settings such as **Splash** screen, game **Icon**, and so on.
Modes

The Modes tab contains all five modes of the editor. They are as follows:

- **Place** mode (shortcut key is `Shift + 1`): Place mode allows you to quickly place your recently placed objects and also Engine primitives such as lights, geometries, triggers, volumes, and so on.

- **Paint** mode (shortcut key is `Shift + 2`): Paint mode (also known as Mesh Paint) allows you to interactively paint vertex colors on Static Mesh in Level Viewport.

- **Landscape** mode (shortcut key is `Shift + 3`): Landscape mode lets you create a new landscape entirely in Unreal Editor or import a height map from an external program, such as World Machine, TerreSculptor, and so on, and make modifications to it.

- **Foliage** mode (shortcut key is `Shift + 4`): Foliage mode allows you to paint or erase multiple static meshes on Landscapes, other static meshes, and so on. An example workflow is to paint grass, trees, and so on on a large area.

- **Geometry Editing** mode (shortcut key is `Shift + 5`): Geometry mode allows you to edit BSP brushes.
Content Browser

Content Browser is what you call the heart of your project. This is where you create, import, view, edit, organize, and modify all the assets for your game. It also lets you rename, delete, copy, and move assets across other folders just like you do in Windows Explorer. Additionally, Content Browser also lets you search for specific assets based on keywords or asset type and you can exclude assets from your search by adding ‘-’ (hyphen) as the prefix.

You can also create Collections to arrange your commonly used assets for quick access.

Collections are just references to assets and are not moved into collections. That means a single asset can exist in multiple collections and you can create an unlimited number of collections.

There are three types of collections:

- **Shared collection**: These are visible to you and to other users. This option is active only if you have Source Control (for example: Perforce, Subversion and so on.) enabled.
- **Private collection**: These are visible only to those who are invited to view the collection. This option is active only if you have Source Control (for example: Perforce, Subversion and so on.) enabled.
- **Local collection**: These are only for you. That means they only exist on your local machine.

If you want to transfer an asset from one project to another, you can right-click on the asset and choose Migrate..., which will copy that asset and all its dependencies to your new project.

Content Browser can be accessed by pressing Ctrl+Shift+F or from the Windows menu on the menu bar. You can also have four instances of Content Browser at the same time.
This can be really useful when you want to move assets to different folders or to preview various assets in different folders.

**Content Browser view options**

View options lets you do the following:

- Change the thumbnail size
- Change the view style
- Modify the 3D thumbnail, and more
View Options can be accessed from the bottom-right corner of Content Browser.

World outliner

World Outliner shows all the Actors within the level in a tree view. Actors can be selected and modified from world outliner. Right-clicking on an Actor in World Outliner will show the same context menu used in Viewport so you can modify it without having to navigate to them in Viewport. You can drag an Actor to another Actor and attach them together.
World outliner allows you to search for a specific Actor. You can exclude a specific Actor by adding - (hyphen) before the search term and you can force a term to match exactly by adding + before the search term.

![World Outliner](image)

**Details panel**
The Details panel shows all the information, utilities, and functions specific to the selection in the viewport. It displays all the editable properties for the selected Actor and provides additional functionality based on the selected Actor. For example, if you select a Blueprint, the Details panel will show everything related to that Blueprint, that is exposed variables, Blutility events, and so on. If you select a Static Mesh actor, the Details panel will show which material was applied, the collision settings, the physics settings, the rendering settings, and more. The Details panel can be locked to the selected Actor so it does not change based on Actor selection. Just like Content Browser, you can have four instances of Details panel open at the same time.

**Navigating the Viewport**
You can navigate the viewport easily using the mouse and keyboard.
A high-level explanation of navigating the viewport can be found here:
https://docs.unrealengine.com/latest/INT/Engine/UI/LevelEditor/Viewports/ViewportControls/index.html
At the bottom-left corner of the viewport, there is a small question mark button. If you click on that, you will see some commonly used viewport shortcut keys.

<table>
<thead>
<tr>
<th>Navigation</th>
<th>Perspective Viewport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move forward / backward</td>
<td>LMB + Drag up / down</td>
</tr>
<tr>
<td>Rotate left / right</td>
<td>LMB + Drag left / right</td>
</tr>
<tr>
<td>Free Rotate</td>
<td>RMB + Drag</td>
</tr>
<tr>
<td>Move up / down</td>
<td>LMB + RMB + Drag</td>
</tr>
<tr>
<td>Zoom in / out</td>
<td>Mouse Scroll Wheel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Top / Front / Side Viewport</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan</td>
<td>RMB + Drag</td>
</tr>
<tr>
<td>Zoom in / out</td>
<td>LMB + RMB + Drag</td>
</tr>
<tr>
<td></td>
<td>or</td>
</tr>
<tr>
<td></td>
<td>Mouse Scroll Wheel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>LMB on Actor</td>
</tr>
<tr>
<td>Toggling selection</td>
<td>Ctrl + LMB on Actor</td>
</tr>
<tr>
<td>Marquee Selection</td>
<td>LMB + Drag</td>
</tr>
<tr>
<td>Clear Selection</td>
<td>Esc</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Focusing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus selected object</td>
<td>F</td>
</tr>
</tbody>
</table>

LMB = Left Mouse Button
RMB = Right Mouse Button
Now that we have some solid understanding of Engine UI, let's use BSP to create a simple level. BSP is a geometry tool (also known as Geometry Brush or simply Brush) used for quickly prototyping levels (also known as blocking out levels). Some developers prefer to call this Constructive Solid Geometry (CSG), which is the more accurate term since geometry in Unreal editor is created by adding and subtracting brushes. BSP has been there since the first release of Unreal. It was used for level designing long ago but later, this role has been passed to static meshes because BSP is more expensive in performance.

So basically, BSP should only be used to prototype a level. Once you have the basic idea of how a level should look, you should start replacing it with static meshes.

Creating BSP

Unreal Engine 4 comes with seven Brushes and all of them can be customized in Details panel. They are as follows:

- **Box**: You can adjust the X, Y, and Z axes and set it to Hollow, which is a fast way to make a room, and adjust Wall Thickness, which defines the thickness of the inside walls.
- **Cone**: You can customize the number of sides, height, and both outer and inner radius in Details panel. You can also set this to Hollow and adjust Wall Thickness to define the thickness of the inside walls.
- **Cylinder**: You can customize the number of sides, height, and both outer and inner radius in Details panel. You can also set this to Hollow and adjust Wall Thickness to define the thickness of the inside walls.
- **Curved Stair**: This creates a staircase shape that bends around an angle but cannot wrap over itself.
- **Linear Stair**: This creates a straight staircase that does not bend.
- **Spiral Stair**: This creates a spiral staircase that can repeatedly wrap over itself.
- **Sphere**: This creates a sphere shape. The radius can be customized in Details panel.
Just like any other actor, you can use **Transform Tools** to move, rotate, and scale as you see fit.

There are two types of **Brushes**. They are as follows:

- **Additive**: These brushes are solid. This will *add* geometry to the level. For example, you will use the **Additive** type to create walls, floors, ceilings, and so on.

- **Subtractive**: These brushes are hollow. This will *subtract* solid space from a previously created **Additive** brush. For example, you will use the **Subtractive** type to create windows or doors on walls.

You can also convert BSP geometry to **Static Mesh** and save them in **Content Browser**, but remember, they will have no UVs or additional Material elements. It is also worth mentioning that this is not a good or recommended workflow. You should only use BSP to block out your level and later, you should import your assets created from a DCC application.

You can go to **Geometry Editing** mode (Shift+F5) to edit vertices and create a custom shape.

---

**Default starting level, splash screen, and game icon**

You can change the default starting level for both the game and editor. For example, for the game you may want the **Main Menu** map as the default and for editor you want another level as the default startup level.

It's easy to set them in Unreal editor:

1. Click on **Edit** in the menu bar.
2. Click on **Project Settings**.
3. Go to **Maps & Modes**.
4. Here, you can change the game and editor default map.

You can adjust Splash screen through Project Settings:

1. Go to Windows sections.
2. Change the Splash screen and the game Icon from here.

![Image of Project Settings]

The default dimensions for Splash screens are 600 x 200 and requires a .bmp image. The game Icon requires a 256 x 256 .ICO file.
Summary
Now that you understand the basics of Unreal Engine, it’s time to import some assets from a DCC application such as 3ds Max, Maya, or Blender. In the next chapter, we will create a simple mesh in 3ds Max and import it into Unreal Engine and go through various options, such as setting up materials, collisions, and LODs.
Where to buy this book
You can buy Unreal Engine 4 Game Development Essentials from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.