Learning Unreal Engine Game Development

Many people know what a game is and they play games every day, but how many of them know how to create a game? Unreal Engine technology powers hundreds of games, and thousands of individuals have built careers and companies around skills developed using this engine.

Learning Unreal Engine Game Development starts with small, simple game ideas and playable projects that you can actually finish. We start off with the basics of using Unreal Engine to create a simple game level. Then, you'll learn how to add details such as actors, animation, effects, and so on, to the game. By the end of this book, you'll have learned how to use Unreal Engine to create your own games.

Who this book is written for

If you are new to game development and want to learn how games are created using Unreal Engine 4, this book is the right choice for you. You do not need prior game development experience, but it is expected that you have played games before.

What you will learn from this book

- Learn what a game engine is, the history of Unreal Engine, and how game studios create games
- Explore the Unreal Engine 4 editor controls and learn how to use the editor to create a room in a game level
- Understand the basic structures of objects in a game, such as the differences between BSP and static meshes
- Make objects interactive using level blueprints
- Learn more about computer graphics rendering
- Get acquainted with the Material Editor to create materials and use different types of light in the game levels
- Utilize the various editors, tools, and features such as UI, the particle system, audio, terrain manipulation, and cinematics in Unreal Engine 4 to create game levels

In this package, you will find:

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

Joanna Lee has more than 8 years of experience in game development. She has designed and programmed various video games. She first started working with Unreal's game engine in 2005 and is very excited to be able to author a book about the newest Unreal Engine 4. She has also worked with many other engines as well as reviewed books and videos on Cry Engine 4.
Preface

Many people know what a game is, and a lot of people play games every day. But how many people know how to create a game? Game development using Unreal Engine 4 allows aspiring game creators to develop professional-looking games quickly. Unreal Engine 4 provides very polished game development tools and capabilities that allow vast amounts of customization for almost any game that you can dream of.

What this book covers

Chapter 1, An Overview of Unreal Engine, covers introductory content about what a game engine is, specifically for Unreal Engine 4 and its history. You will get an overview of the features of Unreal Engine 4 and how it can help you to create a game.

Chapter 2, Creating Your First Level, explains how to create your first room using the Box Brush, add materials to texture the walls/floor, and learn how to place static objects to enhance the look of the room.

Chapter 3, Game Objects – More and Move, covers the structure of a simple object type, known as Static Mesh, and how objects in Unreal interact with one another. This chapter also introduces Blueprints, which is the graphical scripting of Unreal Engine 4.

Chapter 4, Material and Light, shows you how to customize your level in greater detail by learning how to create your own basic custom Material and how to use simple lights to light up the interior of the level.

Chapter 5, Animation and AI, covers how animation works in Unreal Engine and how to implement simple AI in your game level.

Chapter 6, A Particle System and Sound, explains how to add visual and sound effects to your level.

Chapter 7, Terrain and Cinematics, shows you how to add the final touches to your level using terrain manipulation and cinematics.
An Overview of Unreal Engine

First of all, thank you for picking up this book. I am sure you are excited to learn how to make your own game. In this chapter, I will run you through the different fundamental components in a game and what Unreal Engine 4 offers to help you make your dream game.

The following topics will be covered in this chapter:

- What is in a game?
- The history of Unreal Engine (UE)
- How is game development done?
- The components of UE and its editors

What goes into a game?

When you play a game, you probably are able to identify what needs to go into a game. In a simple PC shooting game example, when you press the left mouse button, the gun triggers. You see bullets flying, hear the sound of the gun and look around to see if you have shot anything. If you did hit something, for example, a wall, the target receives some form of damage.

As a game creator, we need to learn breakdown what we see in a game to figure out what we need for a game. A simple breakdown without going into too much detail: link the mouse click to the firing of the bullets, play a sound file that sounds like a gun firing, display sparks (termed as particle effect) near the barrel of the gun and the target shows some visible damage.
An Overview of Unreal Engine

Bearing this example in mind, try visualizing and breaking any game down into its fundamental components. This will greatly help you in designing and creating a game level.

There is a lot going on behind the scenes when you are playing a game. With the help of Unreal Engine, the interaction of the many components has been designed and you will need to customize it for your own game. This is a huge time saver when you use an engine to create a game.

What is a game engine?

What a game engine does is that it provides you with tools and programs to help you customize and build a game; it gives you a head-start in making your own game. Unreal Engine is one of the more popular choices in the market currently and it is free for anyone to use for development (royalties need to be paid only if your game makes a profit; visit https://www.unrealengine.com/custom-licensing for more information). Its popularity is mainly due to its extensive customizability, multiplatform capabilities, and the ability to create high quality AAA games with it. If you intend to start a career in game development, this is definitely one of the engines you want to start playing with and using to build your portfolio.

The history of Unreal Engine

Before explaining what this amazingly powerful game engine can do and how it works, let us take a short trip back into the past to see how UE came about and how it has evolved into what we have today.

For gamers, you are probably familiar with the Unreal game series. Do you know how the first Unreal game was made? The engineers at Epic Games built an engine to help them create the very first Unreal game. Over the years, with the development of each generation the Unreal game series, more and more functionalities were added to the engine to aid in the development of the game. This, in turn, increased UE’s capabilities and improved the game engine very quickly over the years.

In 1998, the first version of UE made the modding of a first player shooting game possible. You could replace Unreal content using your own and tweak the behavior of the non-player characters (NPCs), also known as bots (players that are controlled by the computer through artificial intelligence) using UnrealScript. Then multiplayer online features were added into UE through the development of Unreal Tournament, which is an online game. This game also added PlayStation 2 to the list of compatible platforms in addition to the PC and Mac.
By 2002, UE had improved by leaps and bounds, bringing it into the next generation with the development of a particle system (a system to generate effects such as fog and smoke), static mesh tools (tools to manipulate objects), a physics engine (allows interaction between objects such as collisions) and a Matinee (a tool to create cut scenes, which is a brief, non interactive movie). This improvement saw to the development of the *Unreal Championship* and *Unreal Tournament 2003*. The release of *Unreal Championship* also added the Xbox game console to the list, with multiplayer capabilities in Xbox Live.

The development of Epic’s next game *Unreal II: The Awakening* edged UE forward with an animation system and overall improvement with their existing engine. The development of faster Internet speeds in the early 2000s also increased the demand of multiplayer online gaming. *Unreal Tournament 2004* allowed players to engage in online battles with one another. This saw the creation of vehicles and large battlefields, plus improvements in online network capabilities. In 2005, the release of *Unreal Champion 2* on the Xbox game console reinforced UE capabilities on the Xbox console. It also saw the creation of a very important feature of a new third-person camera. This opened up greater possibilities in the types of games that could be created using the engine.

*Gears of War*, one of the most well-known franchises in the video games industry, pushed Epic Games to create and release the third version of its game engine, *Unreal Engine 3*, in 2006.

The improvement of the graphics engine used DirectX 9/10 to allow more realistic characters and objects to be made. The introduction of *Kismet*, which is a visual scripting system, allowed game and level designers to create game play logic for more engaging combat play without having to delve into writing codes. Platform capabilities of UE3 include Xbox360 and PlayStation 3 was added. There was a revamp in the light control and materials. UE3 also had a new physics engine. *Gears of War 2* released in 2008 saw the progressive improvements to UE3. In 2013, the *Gears of War Judgment* was released.

PC online gaming was also under the radar of Epic Game's developers. In 2009, *Atlas Technology* was released to be used in conjunction with UE to allow *massively multiplayer online games* (MMOG) to be created.

The increasing demand of mobile gaming also led to UE3 being pushed in the direction of increasing its supportability for various mobile platforms. All these advancements and technological capabilities have made UE3 the most popular version of Unreal Engine and it is still very widely used today.

UE3 dominated the market for 8 years until UE4 came along. UE4 was launched in 2014 and introduced the biggest change by replacing Kismet with the new concept of *Blueprint*. We will discuss more about the features of UE4 later in the chapter.
Game development

Each game studio has its own set of processes to ensure the successful launch of its game. Game production typically goes through several stages before a game is launched. In general, there is a preproduction/planning, production stage, and postproduction stage. Most of the time is normally spent in the production stage.

Game development is an iterative process. The birth of an idea is the start of this process. The idea of the game must first be tested to see if it is actually fun to the target audience. This is done through prototyping the level quickly. Iterations of this prototype into a fully-fledged game can go from weeks to months to years.

The development team takes care of this iteration process. Everyone's contribution of the game throughout the development cycle directly affects the game and its success.

Development teams loosely consist of several specialized groups: artists (2D/3D modeler, animator), cinematic creators, sound designers, game designers, and programmers.

Artists

They create all visible objects in the game from menu buttons to the trees in the game level. Some artists specialize in 3D modeling, while others are focused on animation. Artists make the game look beautiful and realistic. Artists have to learn how to import their created images/models, which are normally created first using other software such as 3DMax, Maya, and MODO into UE4. They would most likely need to make use of Blueprint to create certain custom behaviors for the game.

Cinematic creators

Many cinematic experts are also trained artists. They have a special eye and creative skills to create short movie scenes/cut scenes. The Matinee tool in UE4 will be what they would be using most of the time.

Sound designers

Sound designers have an acute sense of hearing and they are mostly musically trained. They work in the sound labs to create custom sounds/music for the game. They are in charge of importing sound files into UE4 to be played at suitable instances in the game. When using UE4, they would be spending most of their time using the Sound Cue Editor.
Game designers
Designers determine what happens in the game, what goes on in the game, and what the game will be about. In the planning stage, most of the time will be spent in discussion, presentations, and documentation. In the production stage, they will oversee the game prototyping process to ensure that the game level is created as designed. Very often designers spend their time in the Unreal Editor to customize and fine-tune the level.

Programmers
They are the group that looks into the technology and software the team needs to create the game. In pre-production, they are responsible for deciding which software programs are required and are capable of creating the game. They also have to ensure that the different software used are compatible with one another. Programmers also write codes to make the objects created by the artist come alive according to the idea that the designers came up with. They program the rules and functionality of the game. Some programmers are also involved in creating tools and research for the games. They are not directly involved in creating the game but instead are supporting the production pipeline. Games with extreme graphics usually have a team of researchers optimizing the graphics and creating more realistic graphics for the game. They spend most of their time in codes, probably coding in Visual Studio using C++. They are also able to modify and extend the features of UE4 to support the needs of the game that they are developing.

The components of Unreal Engine 4
Unreal Engine is a game engine that helps you make games. Unreal Engine is made up of several components that work together to drive the game. Its massive system of tools and editors allows you to organize your assets and manipulate them to create the gameplay for your game.

Unreal Engine components include a sound engine, physics engine, graphics engine, input and the Gameplay framework, and an online module.

The sound engine
The sound engine is responsible for having music and sounds in the game. Its integration into Unreal allows you to play various sound files to set the mood and add realism to the game. There are many uses for sounds in the game. Ambient sounds are constantly in the background. Sound effects can be repeated when needed or one-off and are triggered by specific events in the game.
In a forest setting, you can have a combination of bird sounds, wind, trees, and leaves rustling as the ambient sound. These individual sounds can be combined as a forest ambient sound and be constantly playing softly in the background when the game character is in the forest. Recurring sounds such as footprint sound files can be connected to the animation of the walking movement. One-time sound effects, such as the explosion of a particular building in the city, can be linked to an event trigger in the game. In Unreal, the triggering of the sounds is implemented through cues known as **Sound Cue**.

**The physics engine**
In the real world, objects are governed by the laws of physics. Objects collide and are set in motion according to Newton's laws of motion. Attraction between objects also obeys the law of gravity and Einstein's theory of general relativity. In the game world, for objects to react similarly to real life, it has to have the same system built through programming. Unreal physics engine makes use of the PhysX engine, developed by NVIDIA, to perform calculations for lifelike physical interactions, such as collisions and fluid dynamics. The presence of this advanced physics engine in place allows us to concentrate on making the game instead of spending time making objects interact with the game world correctly.

**The graphics engine**
For an image to show up on screen, it has to be rendered onto your display monitor (such as your PC/TV or mobile devices). The graphics engine is responsible for the output on your display by taking in information about the entire scene such as color, texture, geometry, the shadow of an individual object and lighting, and the viewpoint of a scene, and consider the cross-interaction of the factors that affect the overall color, light, shadow, and occlusion of the objects.

The graphics engine then undergoes massive calculations in the background using all these information before it is able to output the final pixel information to the screen. The power of a graphics engine affects how realistic your scene will look. Unreal graphics engine has the capabilities to output photorealistic qualities for your game. Its ability to optimize the scene and to process huge amount calculations for real-time lighting allows users to create realistic objects in the game.

This engine can be used to create games for all platforms (PC, Xbox, PlayStation, and mobile devices). It supports DirectX 11/12, OpenGL, and JavaScript/WebGL rendering.
Input and the Gameplay framework

Unreal Engine consists of an input system that converts key and button presses by the player into actions performed by the in-game character. This input system can be configured through the Gameplay framework. The Gameplay framework contains the functionality to track game progress and control the rules of the game. Heads-up displays (HUDs)/user interfaces (UIs) are part of the Gameplay framework to provide feedback to the player during the course of the game. Gameplay classes such as GameMode, GameState, and PlayerState set the rules and control the state of the game. The in-game characters are controlled either by players (using the PlayerController class) or AI (using AIController class). In-game characters, whether controlled by the player or AI, are part of a base class known as the Pawn class. The Character class is a subset of the Pawn class, which is specifically made for vertically-oriented player representation, for example, a human.

With the Unreal Gameplay framework and controllers in place, it allows for full customization of the player's behavior and flexibility, as shown in the following figure:

Light and shadow

Light is a powerful tool in game creation. It can be used in many ways, such as to create the mood of a scene or focus a player's attention on objects in the game. Unreal Engine 4 provides a set of basic lights that could be easily placed in your game level. They are Directional Light, Point Light, Spot Light, and Sky Light.
Directional Light emits beams of parallel lights, Point Light emits light like a light bulb (from a single point radially outward in all directions), Spot Light emits light in a conical shape outwards, and Sky Light mimics light from the sky downwards on the objects in the level:

![Light Types Diagram](image)

The effective design of light also creates realistic shadows for your game. By choosing the types of light in the level, you can affect both the mood and time it takes to render the scene, which in turns affect the frames per second of your game. In the game world, you can have two types of shadows: static and dynamic. Static shadows can be prebaked into the scene and, which makes them quick to render. Dynamic shadows are changed during runtime and are more expensive to render. We will learn more about lights and shadows in *Chapter 4, Light and Environment Control*.

**Post-process effects**

Post-process effects are effects that are added at the end to improve the quality of the scene. Unreal Engine 4 provides a very good selection of post-process effects, which you can add to your level to accentuate the overall scene.

It offers full scene **high dynamic range rendering (HDRR)**. This allows objects that are bright to be very bright and dark to be very dark, but we are still able to see details in them. (This is NVIDIA's motivation for HDR rendering.)

UE4 post-process effects include Anti-Aliasing using **Temporal Anti-Aliasing (TXAA)**, Bloom, Color Grading, Depth of Field, Eye Adaptation, Lens Flare, Post Process Materials, Scene Fringe, Screen Space Reflection, and Vignette. Although a game is often designed with the post-process effects in mind, users are normally given the option to turn them off, if desired. This is because they often consume reasonable amount of additional resources in return for better visuals.

**Artificial intelligence**

If you are totally new to the concept of **artificial intelligence (AI)**, it can be thought of as intelligence created by humans to mimic real life. Humans created AI to give objects a brain, the ability to think, and make decisions on their own.
Fundamentally, AI is made up of complex rule sets that help objects make decisions and perform their designed function/behavior. In games, NPCs are given some form of AI so that players can interact with them. For example, give NPCs the ability to find a sweet spot to attack. If being attacked, they will run, hide, and find a better position to fight back.

Unreal Engine 4 provides a good basic AI and lays the foundation for you to customize and improve the AI of the NPCs in your game. More details on how AI is designed in Unreal Engine will be discussed in *Chapter 5, Animation and AI.*

**Online and multiplatform capabilities**

Unreal Engine 4 offers the ability to create game for many platforms. If you create a game using Unreal Engine 4, it is portable into different platforms, such as Web, iOS, Linux, Windows, and Android. Also, Universal Windows Platform (UWP) will soon be added as well. It also has an online subsystem to provide games the ability to integrate functionalities that are available on Xbox Live, Facebook, Steam, and so on.

**Unreal Engine and its powerful editors**

After learning about the different components of Unreal Engine, it is time to learn more about the various editors and how they are able to empower us with the actual functionalities to create a game.

**Unreal Editor**

Unreal Engine has a number of editors that help in the creation of the game. By default, the Unreal Editor is the startup editor for Unreal Engine. It can be considered as the main editor that allows access to other subsystems, such as the Material and Blueprint subsystems.

The Unreal Editor provides a visual interface made up of viewports and windows to enable you to import, organize, edit, and add behaviors/interactions to your game assets. Other subeditors/subsystems have very specialized functions that allow you to control details of an asset (how it looks, how it behaves).

The Unreal Editor, together with all the subsystems, is a great tool especially for designers. It allows physical placement of assets and gives users the ability to control gameplay variables without having to make changes in the code.
Material Editor

Shaders and Materials give objects its unique color and texture. Unreal Engine 4 makes use of physically-based shading. This new material pipeline gives artists greater control over the look and feel of an object. Physically-based shading has a more detailed relationship of light and its surface. This theory binds two physical attributes (micro surface detail and reflectivity) to achieve the final look of the object.

In the past, much of the final look is achieved by tweaking values in the shader/material algorithms. In Unreal Engine 4, we are now able to achieve high quality content by adjusting the values of the light and shading algorithms, which produces more consistent and predictable results. More details about Shaders and Materials will be provided in Chapter 4, Light and Environment Control. The following screenshot shows the Material Editor in UE4:

The Cascade particle system

The Cascade particle system provides extensive capabilities to design and create particle effects. Effects from things such as smoke, sparks, and fire can be created by designing the size, color, and texture of each particle and how groups of these particles interact with each other to mimic real-life particle effect behavior. The following screenshot shows the Cascade particle system in UE4:
The Persona skeletal mesh animation

The Persona animation system lets you design and control the animation of the skeleton, skeleton mesh, and sockets of a character. This tool can be used to preview a character's animation and set up blend animation between key frames. The physics and collision properties can also be adjusted through Physics Asset Tool (PhAT). The following screenshot shows the Persona animation system in UE4:
An Overview of Unreal Engine

Landscape – building large outdoor worlds and foliage

To create large outdoor spaces using the editor, Unreal Engine provides sculpting and painting tools through the Landscape system to help us with it. An efficient level of detail (LOD) system and memory utilization allows large scaled terrain shaping. There is also a Foliage editor to apply grass, snow, and sand into the outdoor environment.

Sound Cue Editor

The control of sound and music is done via the Sound Cue Editor. Sounds and music are triggered to play via cues known as Sound Cues. The ability to start/stop/repeat/fade in or out can be achieved using this editor. The following screenshot shows the Sound Cue Editor in UE4:

Matinee Editor

The Matinee Editor toolset enables the creation of game cut scenes and movies. These short clips created could be used to introduce the start of a game level, tell a story before the game begins or even as a promotional video for the game. The following screenshot shows the Matinee Editor in UE4:
The Blueprint visual scripting system

The Blueprint system is a new feature in Unreal Engine. Unreal Engine 4 is the first engine to utilize this revolutionary system. For those who are familiar with Unreal Engine 3, it can be thought of as the enhanced and improved combined version of the Unreal scripting system, Kismet, and the Prefab functionality. The Blueprint visual scripting system enables you to extend code functionality using visual scripting language (box-like flow diagrams joined with lines). This capability means that you do not have to write or compile code in order to create, arrange, and customize behavior/interaction of in-game objects. This also provides nonprogrammers (artists/designers) with the ability to prototype or create a level quickly and manipulate gameplay without having to tackle the challenges of game programming. A cool feature of Blueprint is that you can create variables like in programming by clicking on the object and selecting Create Variable. This opens up what developers can do without messing around with complex coding.
An Overview of Unreal Engine

To help developers debug Blueprint scripting logic, you can see the sequence of events and property values visually on the flow diagrams as it is being executed. Similar to troubleshooting in coding, break points can also be set to pause a Blueprint sequence. The following screenshot shows the Level Blueprint Editor in UE4:

Unreal programming

The access to Unreal Engine's source code gives users the freedom to create almost anything they can dream of. Functionalities of the base code can be extended and customized to create whatever the game needs to have. Learning how Unreal Engine works from the inside can unlock its full potential in game creation.

Unreal Engine has also incorporated very useful debugging features for the coding folks. One of them is the Hot Reload function. This tool enables changes in the C++ code to be reflected immediately in the game. To facilitate quick changes in code, Unreal Engine has also included Code View. By clicking on a function of an object in the Code View category, it shows you directly the relevant codes in Visual Studio where you could make code changes to the object.

Versioning and source control can be set up for game projects that include code changes.
Unreal objects

Actors are the base class of all gameplay objects in Unreal. For the Actors to have more properties and functionalities, the Actor class is extended to various more complex classes. In terms of programming, the Actor class acts as a container class to hold specialized objects called Components. The combination of the functionalities of the Components gives the Actor its unique properties.

A beginner's guide to the Unreal Editor

This is a quick overview of what we can do with the Unreal Editor. We will briefly touch on how we can use the various windows in the editor to create a game.

The start menu

When starting up Unreal Engine, you will be first brought to a menu window by default. This new start menu is simple and easy to navigate. It features a large tab that allows you to select which version of game engine you want to launch and has a clear representation of the projects you have created. It also provides access to Marketplace, which is a library of game samples that are created by others, which you could download (some free, some paid). The menu also provides latest updates and news from Epic to ensure developers are kept abreast of the latest development and changes. The following screenshot shows the start menu:
Project Browser

After launching the desired version of Unreal Engine, the Unreal Project Browser pops up. This browser provides you with the option to create game levels that have been pre-customized. This means that you have a list of generic levels, which you can start building your game levels with. For those who are new to game making, this feature lets you dive straight into building various types of games quickly. You can have a first-person shooting level and third-person game setup, or a 2D/3D side-scrolling platform level directly in either Blueprint or C++ as the base template. What is so awesome about the New Project tab is that it also allows you to select your target device (PC/mobile), image quality target, with or without the Unreal content included in the startup project. The following screenshot shows the Project Browser:

Content Browser

When the Unreal Editor starts, there is a default layout of various windows and panels. One of them is the Content Browser. The Content Browser is a window where you can find all the content (game assets) that you have. It categorizes your assets into different folders such as Audio, Materials, Animations, Particle Effects, and so on. This window has also the Import button, which lets you bring in game assets that were created using other software into the game. The following screenshot shows the default location of the Content Browser (outlined in green):
Chapter 1

The Toolbar is a customizable ribbon that provides quick access to tools and editors. The default layout includes quick access to the Blueprint and Matinee editors. Quick play and launch game function is also part of the standard ribbon layout. These buttons allow you to quickly view your creation in-game. The following screenshot shows the default Toolbar:

![Toolbar Screenshot]
Viewport

The **Viewport** is the window to the game world so what you see is what is in the game. If you have created a level using one of the options provided in the **New Project** menu, you would notice that the camera has been adjusted accordingly to the settings of that pre-customized level. This is the window that you will use to place objects into and move them around. When you click on the **Play** button in the toolbar, this **Viewport** window comes alive and allows you to interact with game level. The following screenshot shows the **Viewport** window being highlighted in the editor:

Scene Outliner

The **Scene Outliner** contains the list of objects that are placed in the scene. It is only what is loaded currently in the scene. You can create folders and have customized names for the objects (to help you identify the objects easily). It is also a quick way to group items so that you can select them and make changes in bulk. The following screenshot shows the **Scene Outliner** highlighted in the editor:
Modes

The **Modes** window gives you the power to create and place objects into the game world. You can select the type of activity you wish to execute. Select from Place, Paint, Landscape, Foliage and Geometry Editing. Place is to put objects into the game world. Paint allows you to paint vertices and textures of objects. Landscape and Foliage are useful tools for making large scale natural terrains in the game. Geometry Editing provides the tools to modify and edit the object. The highlighted area in the following screenshot shows the **Modes** window:
An Overview of Unreal Engine

Summary
In this chapter, we covered introductory content about what a game engine is, specifically Unreal Engine 4 and its history. We also talked a little about how games are developed and various roles that exist in a game company to help create different components of a game. Then, we covered the different components of Unreal Engine and how we can use these different features to help us make our game. Lastly, we covered the different editors that are available to us to help us customize each of the components of the game.

In the upcoming chapters, we'll be going into the details of the functionalities and features of Unreal Engine 4. In the next chapter, you will be exposed to some basic functions in the Unreal Editor and start making your own game level.
Where to buy this book

You can buy Learning Unreal Engine Game Development 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.