3D game development for the iOS platform
using the Unreal UDK and UnrealScript

Beginning iOS 3D Unreal Games Development

Robert Chin

Apress®

Beginning iOS 3D Unreal Games Development

Robert Chin

Apress®

Beginning iOS 3D Unreal Games Development

ISBN 978-1-4302- 4035-8

ISBN 978-1-4302- 4036-5 (eBook)

President and Publisher: Paul Manning
Lead Editor: Michelle Lowman
Development Editor: Chris Nelson
Technical Reviewers: Thomas Havlik and David Franson
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Copy Editor: Lori Cavanaugh
Compositor: MacPS, LLC
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 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.

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

Apress and friends of ED books 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 Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book’s source code, go to http://www.apress.com/source-code/.

Contents at a Glance

Contents

About the Author

Robert Chin has a bachelor of science degree in computer engineering and is experienced in C/C++ and UnrealScript. He has written 3D games in C/C++ using the DirectX and OpenGL graphics APIs for the Windows platform. He has served as an Unreal UDK consultant and written UDK UnrealScript-based programs for clients, including an entire commercial game coded specifically for the iOS platform.

About the Technical Reviewers

David Franson has been involved in networking, programming, and 2D and 3D computer graphics since 1990. He is the author of *2D Artwork and 3D Game Modeling for Game Artists* (Cengage, 2002) and *The Dark Side of Game Texturing* (Course Technology, 2004). He has also produced digital artwork for 3D video games, film, and television.

Thomas Havlik is the lead developer at Action Mobile as well as the website administrator and a programmer at Raw Games.

Acknowledgments

I would like to thank Chris Nelson for providing critical suggestions as to the direction of this book and for serving as the development editor. I would also like to thank technical reviewers Thomas Havlik and David Franson. I would also like to thank Jennifer Blackwell who helped keep me on track with reminders on when material was due in her role as coordinating editor. Finally I would like to thank Michelle Lowman who helped me get this book contract in the first place by presenting my proposal to the Apress editorial board.

Introduction

The release of the Unreal Development Kit is really the first time a powerful 3D commercial game engine has been available to the masses of ordinary people for free. The underlying technology has been used for numerous high-quality commercial triple-A games that you see in the retail stores in the United States and around the world. The UDK contains the Unreal Engine 3 3D graphics engine and related tools that would normally cost hundreds of thousands of dollars. The only limitation is that the C/C++ source code used to create the UDK is only available to those who pay the full license fee. Thus, you can not modify the UDK engine itself.

This book provides an introduction to using this technology, including the UnrealScript language, for creating 3D iOS games. I have used the technology extensively and used it to create a full commercial physics puzzle type game for iOS similar to the iOS game Angry Birds. It is a powerful tool that is excellent for iOS development. My intention here is to give others a quickstart guide for creating their own iOS games and share game frameworks I've developed that readers can use as the basis for their own work.

Who This Book Is For

This book is for people that want to use the Unreal Development Kit (UDK) to create 3D games for Apple's iOS platform. This includes devices such as the iPhone, iPad, and iPod Touch. This book also is useful for people that want to develop games on the PC platform with the UDK since much of what is covered in this book would apply to creating a game for the PC as well.

This book assumes the reader has some experience with an object-oriented programming language like C++ or at least some programming experience in general. However, several basic game frameworks are presented in this book as a means to help those who are not professional programmers build their own game using the frameworks as a starting point.

It is also assumed that the reader has some basic knowledge of how to use an iOS device since the final game created using the UDK will be played on the actual iOS device.

General Layout of the Book

Before we cover the general layout of this book there are some key points that the reader should note. First, this book is not designed to cover every feature of the UDK since that would realistically involve a set of books, not just one. This book concentrates on the programming side of game development using the default set of assets that come with the UDK. Also, in terms of programming, this book is not meant to provide a full reference to the UnrealScript programming language. This book also isn't intended as a general introduction to iOS development. We have mentioned links to web sites that provide additional useful information throughout this book. Some of the more important ones are listed in the "Other Resources" section at the end of this introduction.

The general format of this book is to discuss UDK topics and then demonstrate these topics in the form of a "Hands-On Example" in which we show you how to develop an UnrealScript program along with the creation of any levels that are needed. We take you, step by step, through these examples along with showing you how to set up any configuration files that are required.

We start with overviews of the UDK and UnrealScript, including a practical coding example. Then we work through key topics with hands on examples and culminating with a complete sample game. Some of these topics rely on 3D math concepts that are reviewed and demonstrated in a separate chapter. Then, in the latter part of the book we present game frameworks which are actually small working games that you can modify and use to build your own custom games. Game frameworks include a physics game, a first-person shooter game, a third-person shooter/adventure game, and a top-down shooter/role playing game.

Other Resources

Epic Games provides a wealth of resources you can use to supplement what you learn in this book:

- Epic's UDK Mobile home page:
  `http://udn.epicgames.com/Three/MobileHome.html`
- Getting Started: Developing Mobile Projects:
  `http://udn.epicgames.com/Three/GettingStartediOSDevelopment.html`
- iOS Provisioning Overview:
  `http://udn.epicgames.com/Three/AppleiOSProvisioning.html`
- Distributing iOS Applications:
  `http://udn.epicgames.com/Three/DistributionAppleiOS.html`
- UnrealScript Language Reference:
  `http://udn.epicgames.com/Three/UnrealScriptReference.html`

Chapter 1

UDK Overview

This chapter covers the basic background information needed to get started with Unreal 3D games development for iOS and for the hands-on examples that follow in subsequent chapters. To start, we take a quick tour of the Unreal Development Kit (UDK) and familiarize those new to Unreal with the development environment. We cover the Unreal Editor, which is where levels are built and assets within the UDK are imported and managed. Some examples of UDK assets are textures, materials, static meshes, skeletal meshes, and sound cues. These are all covered in this chapter. Finally, information specific to game development on the iOS platform using the UDK is also covered. Readers who already use Unreal might want to jump ahead to this section.

Getting Started

The first thing you need to do is go to the UDK's website, located at `http://udk.com`, download the June 2011 Beta version of the UDK (approximately 1.5 GB) that is used in this book, and install it on your computer. The code examples in this book work correctly with the version of the UDK presented in this book at the time of the writing. The UDK is currently still in the Beta phase and new versions of the UDK are being released about every month. After downloading the executable, run the program to install the UDK. At least Microsoft Net Framework 3.1 is required and will be installed on your system if not detected. You can also download UDK Remote at `http://itunes.apple.com/us/app/udk-remote`, which helps with testing your iOS games.

Unreal Editor Overview

Once you have the UDK installed, go to the Start bar and navigate to where you installed the UDK and run the UDK Editor. Once the Unreal Editor is loaded, you should see something similar to Figure 1–1. The Editor is where you build your game levels, as well as manage and manipulate the game assets used in the level. You can run your game on the iOS mobile previewer from the Unreal Editor, as well as set the specific game type to be played.

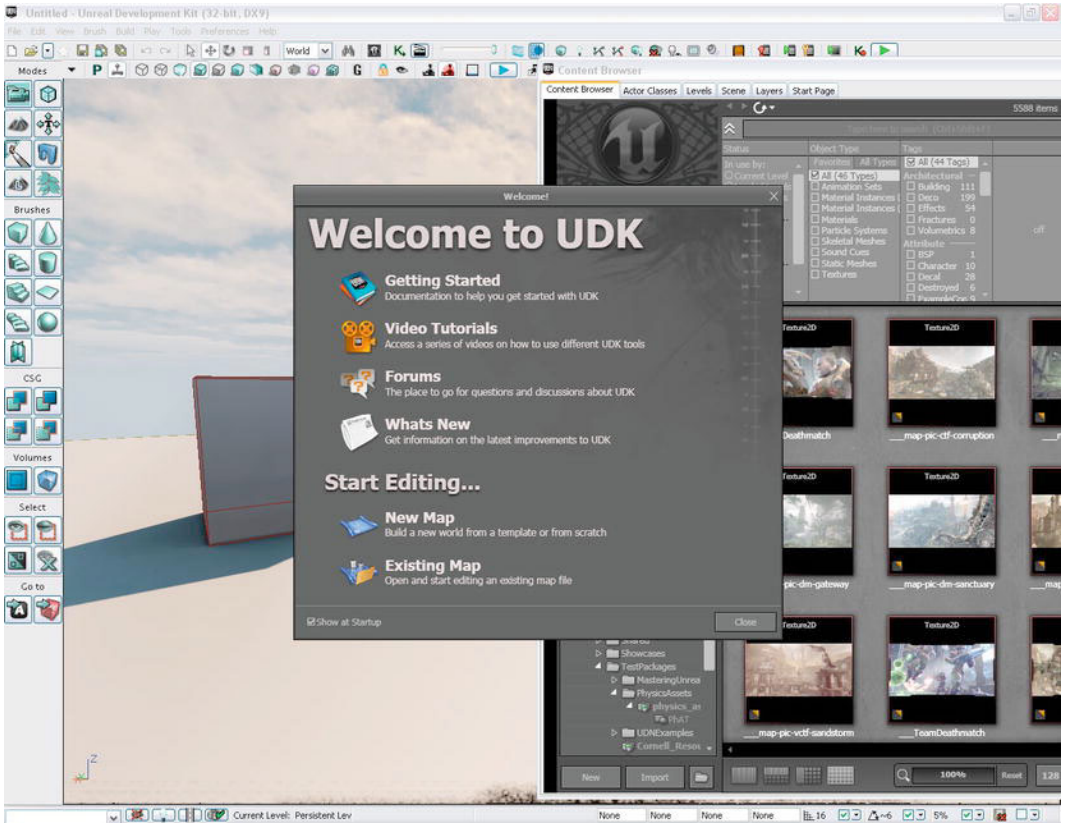

Figure 1–1. *UDK Startup Screen*

Click the Close button inside the Welcome to UDK box to get started. On the right hand side there is a window with many different tabs.

The Generic Browser

I won't go over all the buttons and toolbars in the Unreal Editor UI. We'll discuss all that in context as we work through the book. It is important to take a look at the Generic Browser, however, especially the Content Browser, covered in detail later in this section, and the Actor Classes tab.

As you can see in Figure 1–1, there are six tabs:

- Content Browser. The Content Browser tab is the main interface by which users import, select, and manipulate UDK assets. This tab is discussed in greater detail later in this chapter.
- Actor Classes. The Actor Classes tab contains a list of the UnrealScript classes in the UDK and is subsequently discussed, since it has several elements that will be important early in the book.

- Levels. The Levels tab manages the levels in your world that can consist of one level or many levels that are streamed.
- Scene. The Scene tab displays objects in the current level in table form where you can click on the name of an object and bring up its properties in a side panel.
- Layers. The Layers tab allows you to organize the actors in your level so you can view certain groups of actors and hide others.
- Start Page. The Start Page tab contains internet content related to the UDK, such as documentation, news, community forums, etc.

Now let's take a look at the Actor Classes tab in a bit more detail before moving on to the Content Browser.

Actor Classes Tab

The Actor Classes tab, shown in Figure 1–2, displays the Unreal Script classes currently available. This is where new classes you create appear after you integrate them into the UDK system, as well as classes that are part of the UDK code base.

The term Actor generally refers to an object created from the Actor class or an object created from a class derived from the Actor class. The Actor class is important, because it implements many items needed for gameplay, including code needed for:

- Displaying an object
- Animating an object
- Performing physics and world interaction
- Making sounds
- Creating and destroying the Actor
- Broadcasting messages

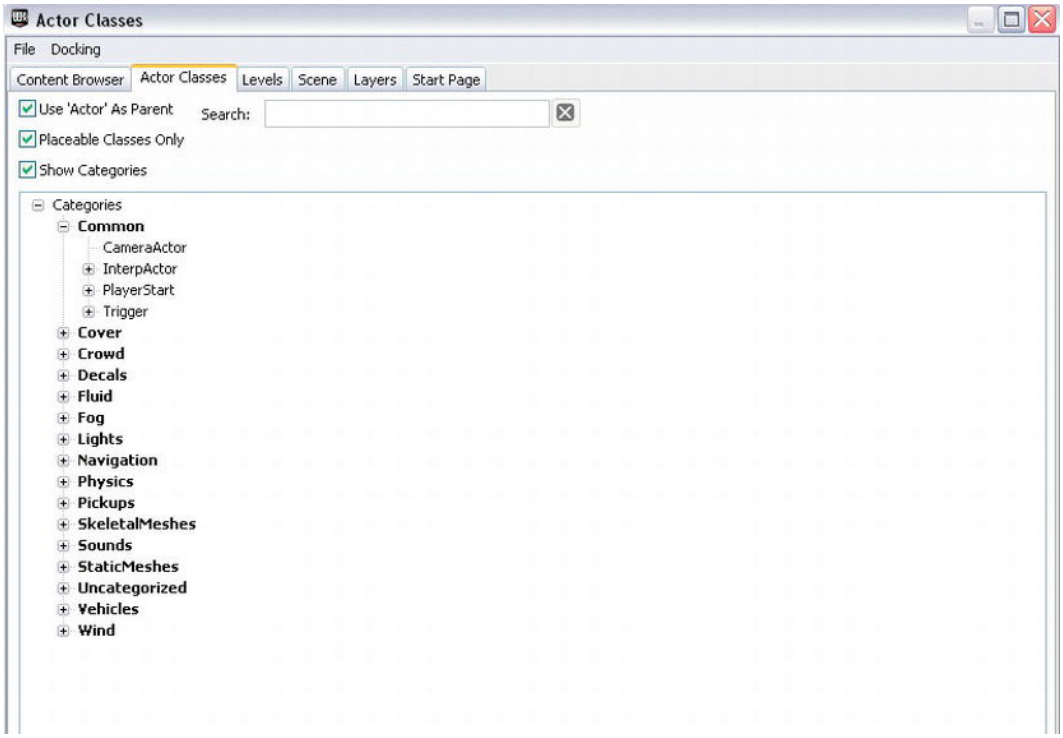

Figure 1–2. *Actor Classes Tab*

There are three checkbox options in this tab:

- Use ‘Actor’ as Parent. Check “Use Actor as Parent” to view only classes that use Actor as a base class. In other words, only classes built from the Actor class. If you uncheck this box, then all classes in the UDK system will be displayed. The class Object will be displayed as the root of the new tree, since Object is the base class of all other classes.
- Placeable Classes Only. If you check the “Placeable Classes Only” checkbox, then only classes that you can place in a game level using the Unreal Editor will be displayed. If you uncheck this box, then both placeable and not placeable classes will be displayed.
- Show Categories. Checking the “Show Categories” checkbox will group and display the classes in different categories like Physics and Navigation.

There is also a search function in which you can search the tree by class name. We use this tab and discuss its features in more detail later in the book.

Now let’s turn to the Content Browser.

The Content Browser and UDK Assets

The Content Browser tab is the starting point for importing and manipulating game content in the UDK system. Game content can be sounds, textures, and 3d computer images used in your game. Click the Content Browser tab to change focus to the Content Browser (see Figure 1–3).

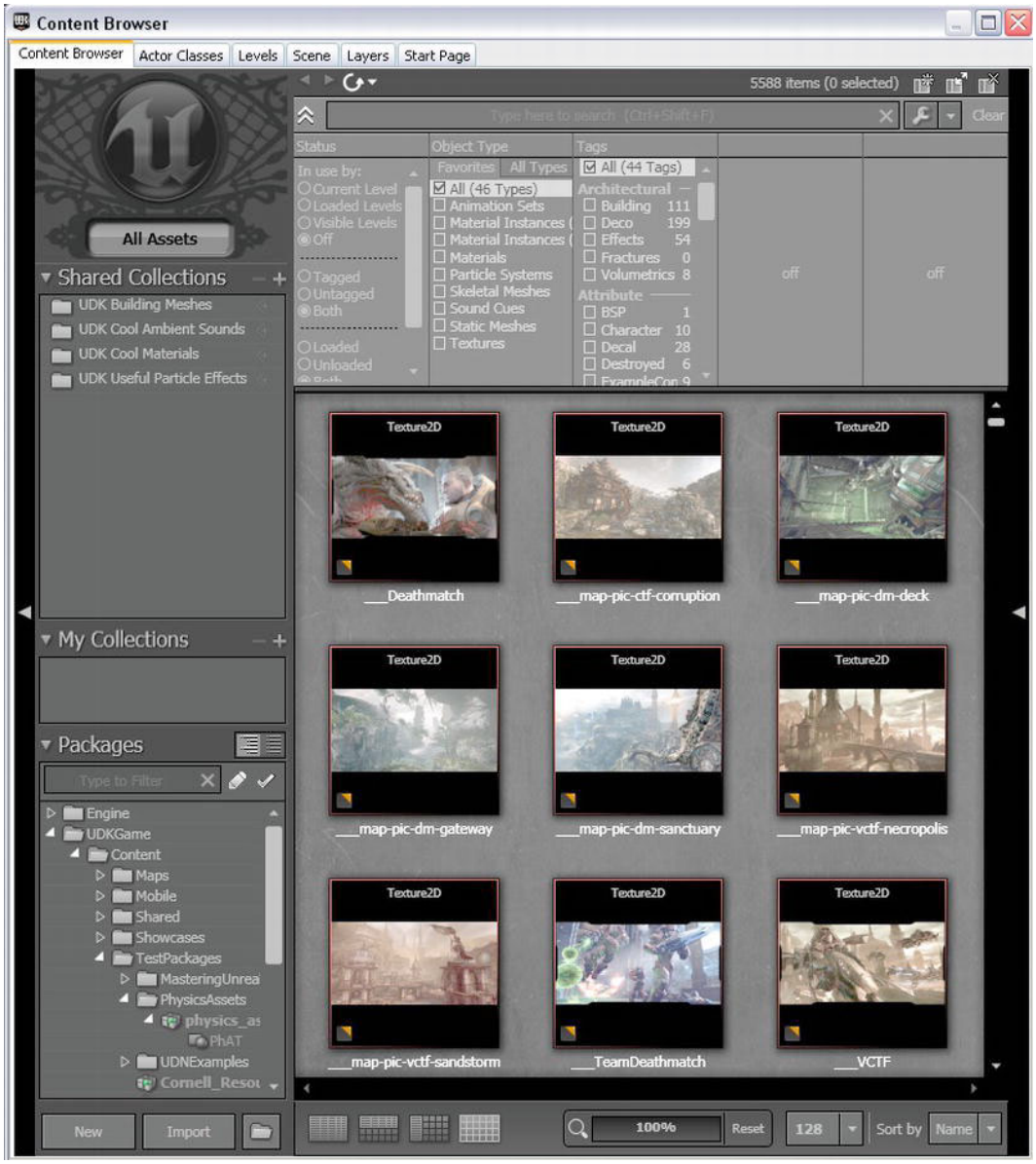

Figure 1–3. *UDK Content Browser*

Importing New Content

You can import new content into the UDK system by clicking the Import button in the lower left hand corner of the Content Browser Tab and can preview that content in the section of the browser where you see the previous images. Clicking the Import button brings up a window in which you can navigate to where your asset is, select it, and then load it into the UDK system. Examples of assets that can be imported from outside the UDK and placed into the UDK system are:

- Sound files in .wav format
- Texture files in .bmp, .pcx, .png, and .tga formats
- Static and Skeletal mesh files in .fbx format
- Movies in .bik format
- Shockwave movies in .swf and .gfx formats

Searching for UDK Assets

You can also filter the objects displayed by name, as well as type. In the upper right side of the Content Browser there is a search box in which you can type the game asset name to search for that is located next to a pair of arrows (see Figure 1–4). There is a section below that with the heading Object Type that contains two subsections named "Favorites" and "All Types." Currently, all of the assets in the game, regardless of type, are displayed, since the "All" checkbox is checked.

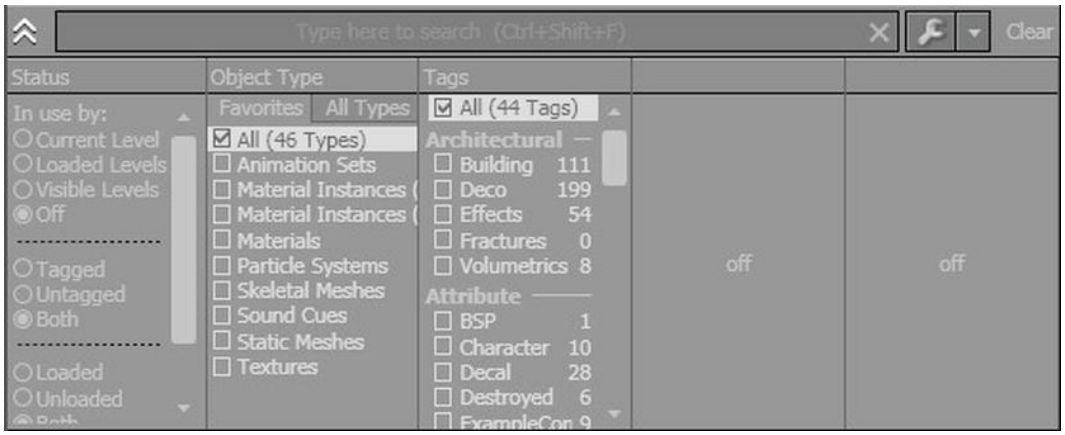

Figure 1–4. *Asset Search Filtering Section of the Content Browser*

Let's search for textures that have "blockwall" as part of their name. Click the Textures checkbox under the Favorites subsection. Next, type in the word "block" to search for textures that contain the word "block" in their name. Finally, under the Packages section of the Content Browser located in the lower left hand corner, select the UDKGame package. Your Content Browser should look something like Figure 1–5.

Figure 1–5. *Searching for Textures Using the "Block" Keyword*

You can double click these texture assets, and a texture's properties window will pop up, giving you more information about each texture asset. For example, click the texture called "T_BlockWall_02_D," and the Texture Properties window shown in Figure 1–6 opens.

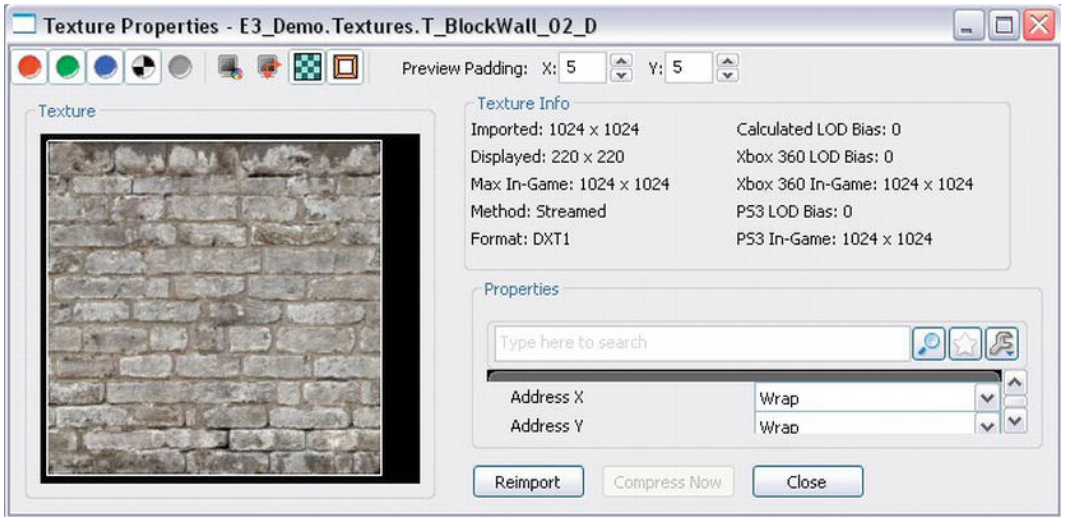

Figure 1–6. *Texture Properties*

UDK Texture Assets

Textures for iOS platforms need to be square. That is, the length in pixels must equal the width in pixels for the texture, such as 512x512 pixels. Textures are generally created outside the UDK system in a paint program like Adobe PhotoShop or PaintShop Pro and saved in a graphics file format, such as windows bitmap (.bmp), that the UDK system can understand and import in. Once inside the UDK system, textures can serve as the building blocks for UDK materials.

Uncheck the Textures checkbox and check the Materials checkbox. Find the material called "M_BlockWall_02_D" and double click it. This will bring up the Unreal Material Editor, and you should see something similar to Figure 1–7.

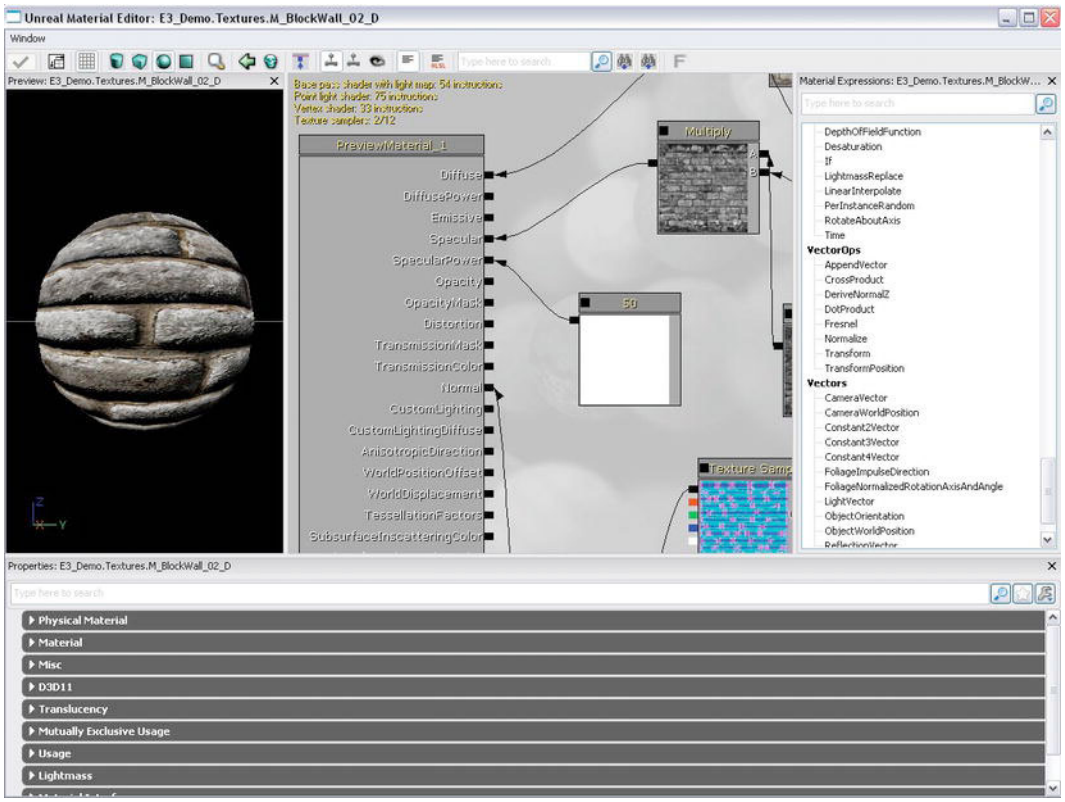

Figure 1–7. *Materials Editor*

UDK Material Assets

The Material Editor is used to create new materials using textures. In the leftmost part of the Material Editor, there is a 3d sphere with a texture applied to it. You can rotate the sphere by clicking it, pressing the left mouse button, and moving the mouse. You can move the sphere forward and backward by clicking it, pressing down the right mouse button, and moving the mouse forward and backward. The texture used for the sphere is the same texture just viewed, which is T_BlockWall_02_D. Verify this is the case by scrolling through the bottom portion of the Material Editor until you come to the Mobile property section. Click the Mobile property if the subproperties are not already displayed (see Figure 1–8).

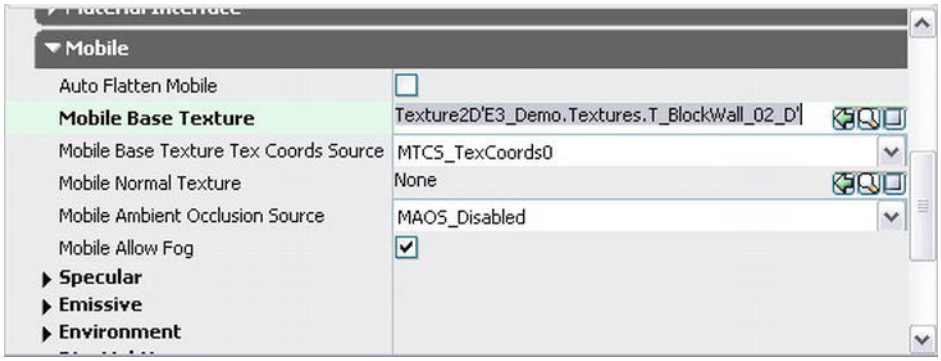

Figure 1–8. *Setting Textures in the Material Editor*

On the right hand side of the Mobile Base Texture property is a set of buttons. These buttons are also used in many other fields throughout the UDK:

- Arrow. The arrow button allows you to select a texture in the content browser, and then click this icon to place the name of that texture here so it can be used as the Mobile Base Texture.
- Magnifying Glass. The magnifying glass button allows you to find the object currently in the field by clicking the icon. When you do this, it should take you to the Content Browser and highlight the texture "T_BlockWall_02_D".
- Clear Screen. The clear screen button clears the Mobile Base Texture property field.

UDK Mesh Assets

A UDK material can be used to provide the surface covering for a mesh, either a static mesh or a skeletal mesh. A mesh is the actual 3d object consisting of a collection of vertices that can be placed in a game level. A skeletal mesh also includes moving parts, called bones, which are generally used to animate a 3d character. The material is what gives the surface of a mesh color and texture.

Now, let's look at an example.

1. Go back to the **Object Type ➤ Favorites** subsection, check Static Meshes, and make sure to uncheck all the other boxes.
2. Type "Cube" into the search box to only display static meshes that have "Cube" as part of their name.
3. Finally, go to the Packages section and click the Engine package. You should see a static mesh called "Cube" in the browser. Double click this item to bring up the Unreal Static Mesh Editor (see Figure 1–9).

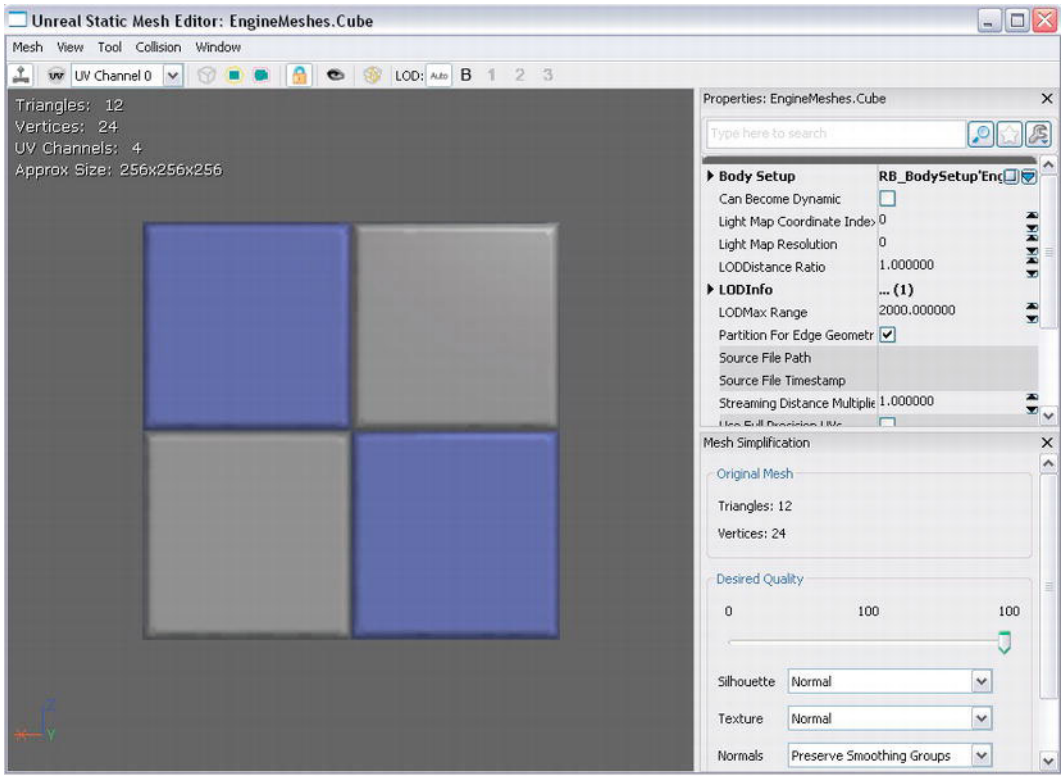

Figure 1–9. *The Static Mesh Editor*

4. You can rotate the cube by first selecting the left hand side of the Mesh Editor that contains the cube. Hold down the right mouse button and move the mouse around to rotate the cube.

5. Hold down the left mouse button and move the mouse back and forth to move the cube view back and forth. Static meshes are meshes without any moving parts.

6. You can view the material this cube is using by going to the LODInfo property section on the right hand side of the viewer, locating the material property, and then clicking the magnifying glass button (see Figure 1–10). This will take you to the Content Browser, and the material used on this mesh will be highlighted.

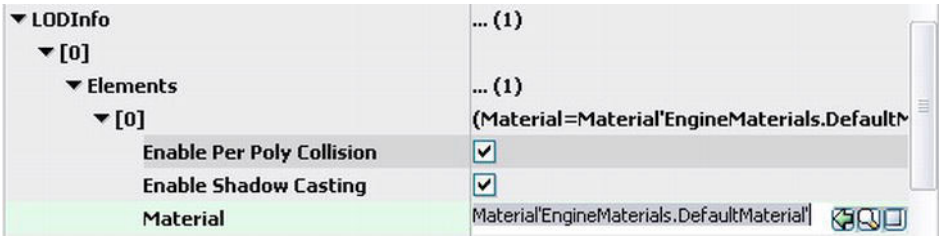

Figure 1–10. *Setting Materials in the Static Mesh Editor*

7. As before, double click the material in the content browser to bring up this material in the Unreal Material Editor.

Now, let's search for skeletal meshes in the UDK. Check the Skeletal Meshes box under the **Object Type ➤ Favorites**, making sure all the other checkboxes are unchecked. Type "Jazz" in the search box and change the Package to search in to UDKGame. You should see a skeletal mesh called "SK_Jazz" in the content browser. Double click this skeletal mesh to bring it up in the Unreal AnimSet Editor (see Figure 1–11).

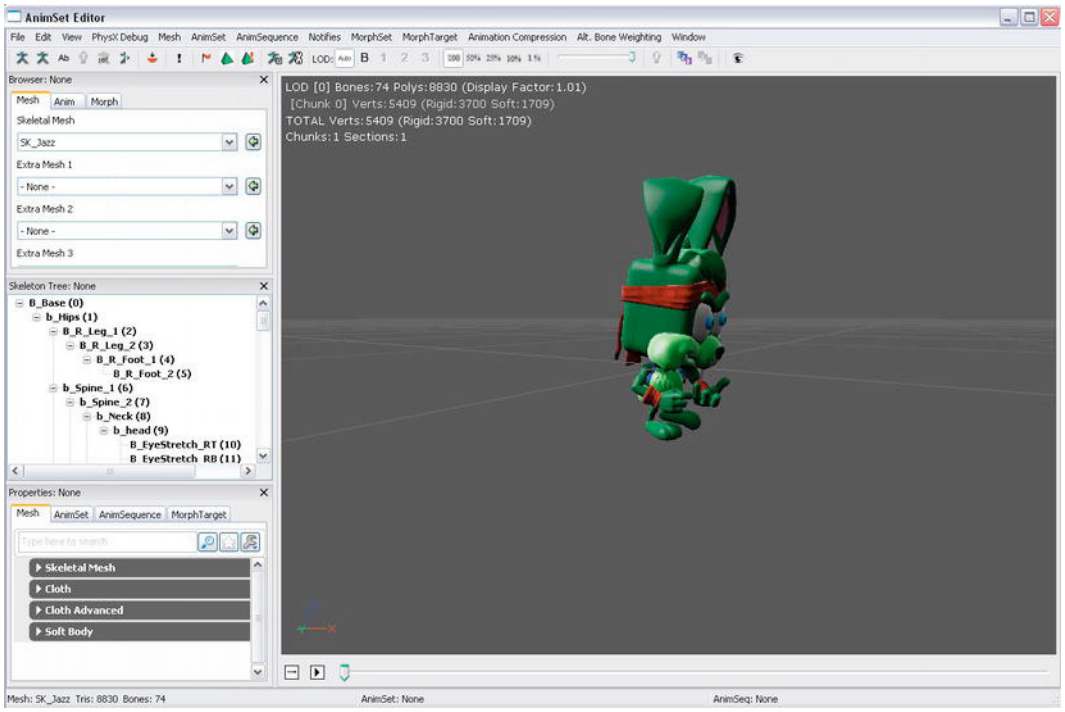

Figure 1–11. *The AnimSet Editor*

You can also set the material for this skeletal mesh. In the lower left hand corner of the AnimSet Editor, under the Mesh tab, you can set the Material property for this skeletal mesh under the Skeletal Mesh category (see Figure 1–12).

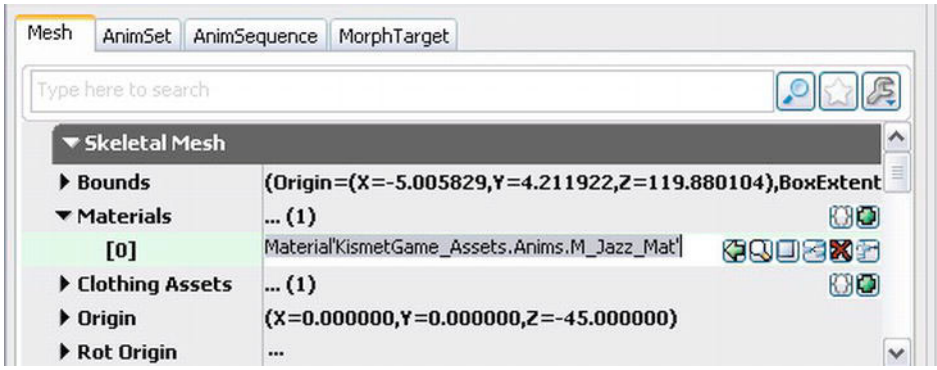

Figure 1–12. *Setting Materials in the AnimSet Editor*

You can also use the magnifying glass button to find the current material in the Content Browser, as well as set a new material from the Content Browser using the Arrow button.

In summary, textures are created in paint programs outside the UDK system and are imported into the UDK system via the Content Browser. These textures can be used to create materials inside the Unreal Material Editor. These materials can then be applied to static meshes via the Static Mesh Editor and skeletal meshes via the AnimSet Editor.

In addition to textures, materials, static meshes, and skeletal meshes, there are two other important game assets within the Content Browser, Particle Systems and Sound Cues.

UDK Particle System Assets

Particle Systems consist of an emitter and the particles that they emit. These are useful for such things as explosions and trails that projectiles leave when fired.

Let's take a look at one.

1. In the Object Type subsection, select Particle Systems as your object type, making sure all the other options are unchecked.
2. Type "fire" as the search filter term, making sure the UDKGame package is highlighted in the Packages section of the Content Browser.
3. Double click the fire particle system displayed to bring up Unreal Cascade, as shown in Figure 1–13.

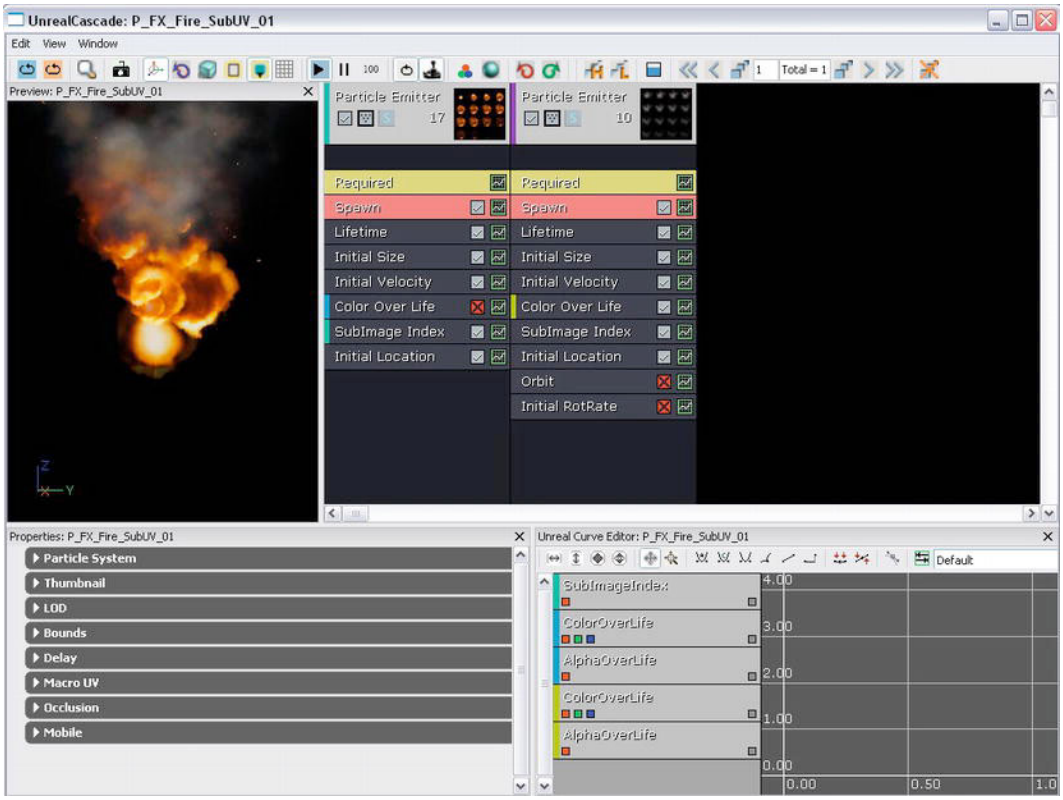

Figure 1–13. *Unreal Cascade*

Unreal Cascade has many options for creating your own custom emitters. Such things, including particle type, particle speed, and particle direction, can be customized. For now, let's not get into the details, but just know that custom emitters can be easily created from within the UDK system.

UDK Sound Cue Assets

Now, let's search for sound cues. Select Sound Cues as the Object Type you will search for by checking its box. You can double click a sound cue to hear it. You can also edit the sound cue in the Sound Cue Editor by right clicking the Sound Cue you want to edit and selecting the "Edit Using Sound Cue Editor" option (see Figure 1–14). This should bring up the Sound Cue Editor shown in Figure 1–15.

> **NOTE:** You can also access the editor for other game assets like static meshes, materials, etc. by right clicking that asset and selecting "Edit Using EditorType". The EditorType will depend on the asset, such as "Edit Using Material Editor" if the asset selected is a material.

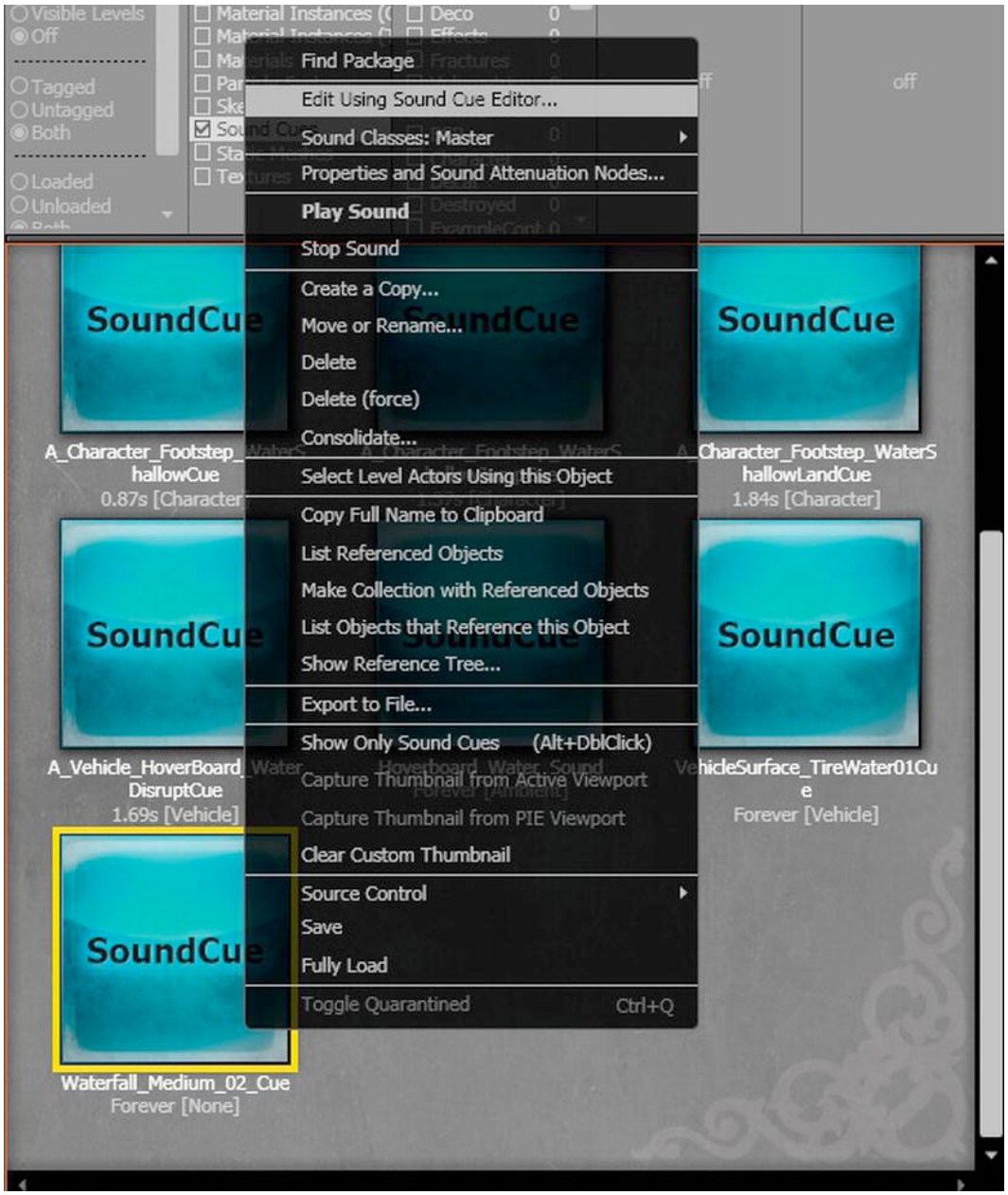

Figure 1–14. *Selecting the Sound Cue Editor*

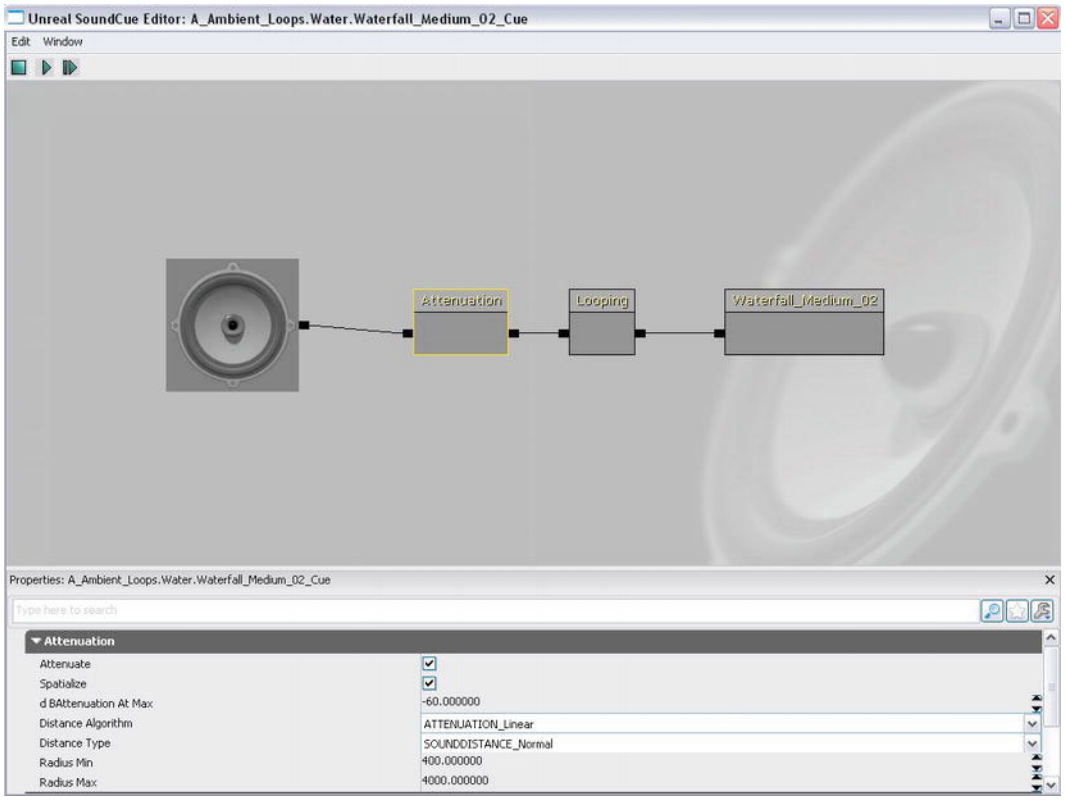

Figure 1–15. *Sound Cue Editor*

The Sound Cue Editor allows you to mix different sound samples into a single sound cue. For example, the sound editor has options for looping a sound and generating a random sound from a group of sounds.

IOS Specific UDK Information

There are certain differences to keep in mind when developing game for the iOS platform. The major differences involve saving data, preparing textures for an iOS device, and the types of player controls available to the user. We will return to the information discussed in this section later in the book and use it in the numerous hands-on examples.

Saving Data on an iOS Device

Some ways of saving data through the UDK system work on the PC-based game and even on a game on the Mobile Previewer but not on an actual iOS device. For example, using config files to save data will work on a PC-based game and even on an iOS-based game using the Mobile Previewer but will not work on the actual device. The best way to