Programming for Artists and Designers

Lecture 1: Introduction to the Unreal Engine/Script

Introduction

This module aims to provide you with more programming experience and to give insight and experience in to a game scripting language. Essentially we will be modding the original game.

Links

Mods: http://www.moddb.com/
Design Goals:
http://wiki.beyondunreal.com/wiki/UnrealScript_Language_Reference/Introduction

What is Modding?

Modding is a slang expression for the act of modifying a piece of hardware or software to perform a function not intended or authorized by the original manufacturer.
When we mention modding it reference to games it can be anything that you do to change something about an existing game. For example, some games allow you to design and play your own levels and make your own player models, while others allow you to invent completely new types of games by modifying the original program code and writing your own code.
We can mod unreal to the extent that a new game, with the exception of the engine, can be created. Generally mods involve the production of new gametypes, levels, maps or weapons.

What is a Game Scripting Language?

The process of using an interpreted language, from a "script" file which is normally text, rather than a compiled executable which is binary. Scripting allows for higher level functions which can be changed without having to rebuild an executable.

Computer languages are created for varying purposes and tasks — different kinds and styles of programming. One common programming task is known as scripting, or connecting diverse pre-existing components to accomplish a new related task. Those languages which are suited to scripting are typically called scripting languages. Many languages for this purpose have common properties: they favour rapid development over efficiency of execution; they are often implemented with interpreters rather than compilers; and they are strong at communication with program components written in other languages.

Many scripting languages emerged as tools for executing one-off tasks, particularly in system administration (and increasingly games). One way of looking at scripts is as "glue" that puts several components together; thus they are widely used for creating graphical user interfaces.

It must be noted however, the boundary between scripting languages and regular programming languages tends to be vague, and is blurring ever more with the emergence of new languages and integrations in this fast-changing area.

What does this mean for us?

This means that the process of modifying the game become much easier, we can access and create scripts with little effort, allowing us alter many variables of the game world.

Game Engines

In computing, a game engine is the core software component of a video game. It typically handles rendering and other necessary technology, but might also handle additional tasks such as game AI, collision detection between game objects, etc. The most common element that a game engine provides is graphics rendering facilities (2D or 3D).

Engines that only provide real-time 3D rendering are sometimes called 3D engines. Such engines include Genesis3D, Irrlicht, and Ogre. Many 3D engines are designed for other purposes, often in addition to games.

Unreal Engine Core

At its simplest, a computer game is made up of the following things:

- A collection of things that describe the environment the player is supposed to experience
- A means of conveying that to the player on her monitor, through her speakers and so on

The collection of things is staggeringly huge, even for the average game of UT2003. In programming terms, each thing is called an object. The Unreal engine is constantly aware of where these objects are in the fictional world, what they are doing and so on. It does this by looking at each one in turn.

So far, we have a long list of objects, and each one is assigned a position in 3-dimensional space using Cartesian coordinates. Now the engine also knows where the player is in the world (the player is represented by just another type of object), and can calculate what the player is supposed to see and hear. If a rocket just flew past in front of you, then you'll see a rocket in front of you on the screen.

The virtual machine

We now come to a fundamental difference between the Unreal Engine and many other first-person shooters, although with the current fashion of game engines being independently licensed that is changing.

It's a difference which, in the opinion of many modders, makes working with Unreal much easier than with other engines.

The parts of the software that handle display on the screen (rendering), sound, and knowing where all these objects are, networking (clients and servers sending each other information about the objects) as well as anything else that is platform-specific, is written in C++, is known as "native". You can't see the code for this (it's "closed-source"), and very rarely is any of the work done by modders involve modifying this.

The native software also creates a virtual machine, similar to the Java virtual machine. This is like a programming environment that sits on top of the native stuff. The language for this is UnrealScript; it's similar to Java and JavaScript. It is compiled into bytecodes, like Java, but the source remains visible, and it's this that allows the modding of the Unreal engine games.

In very simple terms, every single object in the game is a piece of UnrealScript code. The engine runs all these scripts, and they decide what to do. So when the rocket hits a wall, a function in its script called something like HitAWall says: "I have hit a wall. I should now produce an explosion, make a noise, and remove myself from the game." How does the rocket know it hit a wall? The native part of the engine told it. How does the engine know? Every single moment of the game, it is checking where objects are and what they are touching.

The Tick
The engine divides time into ticks. A tick is not a specific length, which allows the engine to scale this according to the power of the machine and the current load. While the C++ parts of the engine are fast, UnrealScript is a relatively slow language to execute. The software is designed in such a way that things that happen all the time, like movement and checking whether objects are touching, is handled natively, while particular events are handled in UnrealScript. While the rocket is moving, no UnrealScript is being executed. The native engine knows to keep moving it forwards each tick, and the renderer is of course showing it to you in a different position each frame (the renderer works in frames, like animation. Again, how fast these go, the FPS, depends on your hardware and how much is going on in the game.)

But when the rocket hits the wall, things begin to happen in UnrealScript:

1) The rocket is notified. This starts a chain of events:
2) The rocket informs the thing its hit: "you've just been hit by a rocket. This is maybe going to hurt you". The thing it's hit could be a player which received damage, or a piece of architecture which won't care much. But the point is that the rocket doesn't care what the target does about being hit. This is a very important point about object-oriented programming, which we'll see later on. Roughly speaking, this is encapsulation.
3) The rocket creates a new object: an explosion. This explosion object will begin to run its own script, which says:

   a. Display bright flashes at your position.
   b. Make a loud noise.
   c. Now some smoke. When the renderer inspects all the objects the player can see, it will see that there are bright flashes and smoke called for at a particular spot.
   d. Then remove yourself from the game, you're done.

4) The rocket then removes itself from the game. The object that represents the rocket no longer exists.

Now from the point of view of the C++ native code, each object is taken in turn. But from a programmer's point of view, we can imagine the rocket and the explosion as running in parallel.

What will we be using?

We will be modding UT2003 using unreal script. For that are interested UT2003 uses the unreal two engine.