back <- up -> next

Unreal Tournament Architecture Overview

In UnrealTournament, all of the things in the world that you interact with, from Bots to elevators to bodies of water, are represented by hierarchical structured objects. The base class, like Java, is object. From this class, all Objects get the nominal amount of data and behavior for their use in the UnrealTournament environment. Examples are UWindow, UT's windowing system, Bitmap, a reference for externally stored bitmaps, and Time, a reference for the system's clock. These are all Objects that are independent of a map and may interact with UT outside of a given map's environment. Another class that extends Object is Actor. Classes that sublass from Actor are anything that will interact with the virtual world once a map has been initiated and the user becomes immersed in the 3D environment. Not all of these are visible or tangible. GameInfo, for example, defines the specifics of the gametype that will be played (DeathMatch, CTF, etc...). However, Actors are all relevant during execution of running map, so they are given special capabilities that are not present in base Objects, like physics, renderability, and network replication. With the possible exception of UWindow, writing a Mod will be done mostly through interaction with those classes that subclass from Actor. I will briefly introduce some of the more relevant Actors and expound upon them in later documents as they become relevant in certain contexts.

- **Pawns** - These are actors that move and behave like living creatures. They have special characteristics like intelligence and the ability to move. You will manipulate the action and AI of NPC's by subclassing from ScriptedPawn or Bot. The former is a generalized class for any sort of NPC that will appear in your game. Examples are animals and the enigmatic Nali. Bots are a branch of pawn with special AI and behavior that is specifically catered to UT. If you want to stray very far from the behavior of the Bots in UT, I suggest using ScriptedPawn. Another important child class of Pawn is PlayerPawn.
- **PlayerPawn** - PlayerPawn is responsible for the Player's avatar, or in-world character. The primary difference is between PlayerPawn and other Pawns is the lack of AI specific code and the addition of interface specific code for display and input, as well as the networking differences which are a result of this transformation from AI driven to interface driven behavior. In a full conversion Mod, you will likely spend a lot of time manipulating a subclass of PalyerPawn. This could be anything from keyboard mappings to the way the player's avatar moves. Another thing that is contained in the PlayerPawn class is a reference to the HUD class with which it is associated.
- **HUD** - The HUD or "Head's Up Display" is a term that comes from flight simulators for all of the information that is displayed on the panel in front of pilots. The idea is that anything that needs to be displayed for the user's benefit that does not exist in the 3D world can be printed to this

translucent panel. Another way to think of it is the in-game windowing system. Note that it is different than the UWindow windowing system, as this used for system control. The HUD is associated with things in the virtual world: ammo, targeting, playermessages, ... and the UWindow system is for interfacing with things outside of the virtual world: Changing Video modes, Exiting UT,... You will almost certainly want to make a new HUD for your game, as this decides a lot of the look and feel, not to mention the interface capabilities, of your Mod.

- **Info** - This class does exactly what it says, it holds information for various things. Most of the subclasses of Info are for internal use, but there are two which are extremely useful and another two which I could conceive might be used in a full conversion. The two essentials are GameInfo and ZoneInfo. The first is the superclass of all game types, including CTF, Assault, and DeathMatch. This is where the rules of your game, or subgames if you have more than one, will be found. If you decide to do an adventure game, where you pick up treasure or garner experience to win, you will put the rules here. ZoneInfo is a marker to describe the environment of a zone (see UTEd tutorial), like water or lava, etc... You may find this useful for changing the weather or environment of your Mod. The two useful, but complex Info subclasses are InternetInfo and ReplicationInfo. Unless you plan on changing or extending the network architecture of UT, you won't need to mess with these.

- **Inventory** - This is the class for items which can be handled, moved, picked-up, etc... in the 3D environment. What makes these different from decorations is that they have special capabilities so that Characters can pick them up. Here you fill find the Weapon subclass as well as PickUp which has things like health and armor. Anything that is a movable object that can be picked up will be subclassed from Inventory. If it cannot, use decoration, its less computationally expensive. Every Inventory object has a pointer to another Inventory object, so more than one can be carried without using an array.