UnrealScript

By: Akbar Ali (syedali011@earthlink.net)

Revision #3.01

Disclaimer:

This article is not some magazine article that is easy to read, or does it have perfect grammar etc. If you are looking for a paper that can get you started coding UnrealScript as fast as possible then this is your paper. If you find any errors then please e-mail me at syedali011@earthlink.net

Assumptions:

In this paper I am assuming that you are a hard working individual who is willing to learn UnrealScript and that you now a little about c programming. Even though UnrealScript is thought of as an object oriented programming language; I will do a quick primer on this for you. This paper is not meant to be a manual but a guide to get you started into coding some UnrealScript. If you are looking for a manual type read then I suggest you visit unreal.epicgames.com and look through the side bar for some papers. A lot of tutorials on the net are dedicated to only helping you understand certain objects and classes. I hope to portray to you in a wide way about how everything gets linked etc. I had spent countless hours trying to figure out UnrealScript but failed miserably due to a few file changes that were not mentioned in the SDK etc.
Well get your coffee ready causes this is going to be a long one.

Beginnings:

Yes like I said in the assumptions I am assuming you no nothing about computers or games just a hard worker that is willing to learn what and how to use the “UnrealScript®”. Well before we start getting into UnrealScript and Unreal Tournament I would like to give you a brief primer on games and the way they work. If you are already familiar with this I suggest you skip this section.
First things off a computer can only read instructions, when you play a game or load up your word-editing program all you are doing to the CPU is

giving it instructions to feed on the data that you give to it. A game is the same way; it is only a bunch of instructions that the CPU will interpret for it to understand and then give you back results in a human readable form. Unreal Tournament is no exception. Know that we have done a little history; let's get back to programming languages. Back in the day a few years ago almost all games were written assembly this was due to the fact that this was the fastest possible means to get your game running at reasonable speeds (fps).
Remember: "game programmer's are always looking to get more from the hardware then it can actually deliver"
Now with CPU's speeds raising the bar each new quarter it now seems that we are not limited buy the hardware but the actual software that we code. In fact, last years supercomputers can be purchased through mail order for about 6,000 dollars now. Due to this fact Epic games introduced "UnrealScript", a type of higher-level language. This was to a full-fledged language to make the code simpler to create and prevent the coder to have to deal with all the nitty-gritty details that are really just fuss. With CPU's speeds on the rise this seemed rather logical as the future will progress we are just going to keep seeing higher and higher level languages cause now we have the CPU power to do this. Before it was needed to code in assembly but now that is just horrible dream that has faded away. Know days you can pick up a very fast computer that can execute instructions at unbelievable speed so it is now no longer needed to optimize in assembly, at least most stuff having to do with game programming. Epic had realized that creating an "UnrealScript" was worth the performance loss due to the fact that the code would be much easier to understand and would allow programmers to concentrate on the game play on not other issues.

UnrealScript

Well you are probably like. "Uhh okay that is nice but still what is UnrealScript" well let me explain be patient. UnrealScript is a language, which is very similar to c ++ and JavaScript. You code the UnrealScript syntax into a text files with the extension of .uc, just as you would a c++ program into a .cpp files. After you have created your text (.uc) files you need to allow UCC to compile these files. Ucc is a compiler that will take your .uc files and compile them, after doing this it will create a .u file which the game it self will use.
I have left a lot of things out but this will be explained later on.
'Ucc Make' is what you would type in your *UnrealTournament->system* directory for you to start building some .u files'.

Inheritance:

A discussion about UnrealScript is bound to head to one thing and that is inheritance. No, No not that inheritance but another definition. In programming terminology you can define inheritance as something that regains all the attributes that where in the previous value. Let me clarify on this. Suppose you have an object called a human, this object will contain all the basic body parts needed to function properly but no specifics (amount of legs, hands, arms, etc). Now we will make a sub-object (sub-class in UnrealScript) and we will call this male. Below is the example on the way you would do this in UnrealScript syntax. Looks very familiar to c++ code.

```
class male expands human
```

What this does is exactly what it says. It is making a class called male that expands on the data that is inside human.

UnrealScript Syntax

For me to write a paper or as you could say a section about the scripts syntax is beyond the scope of this paper. You can find a well-written article by Tim Sweeney called "UnrealScript Reference" which is located at the technology site at unreal.epicgames.com. I feel that Tim has done a good job with this but it is a rather technical document and is not candy coded so be prepared to take a lot of time with that white-paper. I suggest you read and then come back.

What's next

By now you should have a pretty decent understanding of the whole way Ucc works with it's parsing and loading etc. But you have still not understood how the whole entire game loads up. Unreal and Unreal Tournament our the first of there kind in that there game does not rely on a gamex86.dll file to load all the settings variable settings and the way the function work and are implemented.
Unreal and Unreal Tourney use .u files instead of dynamically linked libraries.
Remember those .u files are built by UCC by parsing the .UC files that you build.

Okay in Unreal based games the engine relays on packages. These are things you could say are like a briefcase. Inside are all the necessary details. Like in a briefcase a person might have his wallet his clothes toothpaste etc. Now you have the actual engine. When the game loads it looks in it's UnrealTournament->system directory for all it's necessary and important data to load and get the game running.

Int files

Well if you go thorough your system directory you are bound to see some .Int files. These are what the engine will read in to do it's underlying work. Those files are like the instructions on what packages to load and where to load them. Int files can be opened with a text editor like notepad, WordPad etc. So as you can see ladies and gentleman you could in theory state that the INT files are the heart of the engine.

Ini Files

Yes you read right. There are Ini files. These are again are "**VERY**" important. For your case being a mod developer and all the most important .ini file would be UnrealTournament.ini file. Ucc looks in here when compiling. Basically what UCC does it that it relays on this file to know what to compile it checks in the section that start of with EditPackage. This is followed by a '=' the name that goes after is the folder that ucc will goto and compile the UC files in the =folder->classes folder. I will clarify this for you later on but for right now just know that this is how these files are very important to you.

```
EditPackage
```

Section.

Ucc:

Whenever you think that Ucc is having a problem or something of that nature it is usually due to the fact that your unrealtournament.ini does not contain the name of the folder which you have added new files too.
It should be like this. In the unrealtournement.ini, remember this is what Ucc looks through so it knows what to compile.

```
EditPackage=YourMod.
```

Ready for the beef:

Well buy know you are like okay. Okay I am ready for this script part. Well hear you go. First things first on your unreal tournament go open the second CD. You have to buy the game you know, heheh. Follow these steps and everything will work out. There is other ways you can do this but this is the simplest. I will explain what you are doing later. "*Follow these word for word*"

1. Go to 2nd CD and go to UnrealEd folder
2. Double click on UnrealEdInstaller
3. Unreal Ed is installed
4. Go to UnrealTournament->system Directory
5. Double Click on UnrealEd.exe
6. There will be panel on your right side of your screen. In the bottom right corner there is a button called "Export All". Hit this
7. Know Quit UnrealEd
8. Go to your UnrealTournament directory
9. Now create a folder, this is where you will all your programming files will go into (.uc) Let's make it "mymod" in this case.
10. In "mymod" create a subfolder called "classes".
11. Go back to your UnrealTournament->system directory and create a file called "mymod.int"
12. In your UnrealTournament->system directory, edit your "***UnrealTournament.Ini***" file to say EditPackages=mymod in the section where you will see a bunch of EditPackages stuff. It will look something like this.
    - EditPackages=Core
    - EditPackages=Engine
    - EditPackages=Editor
    - EditPackages=UWindow
    - EditPackages=Fire
    - EditPackages=IpDrv
    - EditPackages=UWeb
    - EditPackages=UBrowser
    - EditPackages=UnrealShare
    - EditPackages=UnrealI
    - EditPackages=UMenu
    - EditPackages=IpServer
    - EditPackages=Botpack
    - EditPackages=UTServerAdmin
    - EditPackages=UTMenu
    - EditPackages=UTBrowser
    - EditPackages=mymod

There you have it, it a matter of 12 easy steps you have created a workspace where you can code in. Know time for the juicy details where I explain what you did.

Step 1:
You go to the second CD because that is where your UnrealEd.exe is located. This is the application that will dump all the .uc files that were created to make the game.

Step 2:
Tell windows to execute the application.

Step 3:
Verification to the user.

Step 4:
Current location to move to, these really aren't steps you know.

Step 5:

(again) To execute the application and which app to click into. UnrealEd folder.

Step 6:
Where the button is located. This is very important. If you don't do this, then all the .uc files that are your Software Development Kit will not be dropped (exported).

Step 7:
Leave UnrealEd it is no good if you want to be a programmer. I suggest you use dev Studio to handle your .uc files to code. You can use notepad or any ASCII text manipulation program. Including emacs if you will ☹

Step 8:
Switch directory

Step 9:
Under your c: UnrealTournament directory add a new folder like "mymod". So your directory structure looks like this now
C:\UnrealTournament\mymod

Step 10:
Go into the UnrealTournament->system directory and create a new file and call it mymod.int. This file is used by Unreal Tournament engine to load up your settings and your changes etc. This is important also when you create your mods. All your changes that you have made must be stated to UnrealTournament. This is done by using that file as the interface. Look at Botpack.int in the same file structure. Your int files will start too look like that once you start coding a lot of stuff. Hope fully you will not have downtime for a long time.

Step 11:
See Above.

Step 12:
That is what you edit, and where you edit it.

Try it:

Now that you have made your workspace and code environment ready test it out. Know you have already accomplished the most difficult task that was setting up the entire setting. Try out your newfound compiler now. Go to your Unreal->system from the dos prompt. Know type Ucc make. You should see a whole bunch of stuff coming up on the screen. Here is an example of what you should be getting.

```
--------------------Core--------------------
--------------------Engine--------------------
--------------------Editor--------------------
--------------------UWindow--------------------
--------------------Fire--------------------
--------------------IpDrv--------------------
--------------------UWeb--------------------
--------------------UBrowser--------------------
--------------------UnrealShare--------------------
--------------------UnrealI--------------------
--------------------UMenu--------------------
--------------------IpServer--------------------
--------------------Botpack--------------------
--------------------UTServerAdmin--------------------
--------------------UTMenu--------------------
--------------------UTBrowser--------------------
--------------------mymod--------------------
Success - 0 error(s), 0 warnings
Tool returned code: 0
```

Well there you go you have compiled all those files that you had exported “dumped” into your UnrealTournament directory. I did not mention it but when you went to UnrealEd and hit export all, all you did was dump all those .uc files into your UnrealTournament Directory. Those files went into their designated place. For example open up folder Botpack. Daja vu? “Your folder” should look the exact same with another class folder in the inside. Now click on the class folder you should many .uc files these are all the files that UCC compiles to make the games BotPack.u. You don’t ever modify these files just use those files for reference, or to mimic what they do in your mod etc. Now goto your folder “mymod” switch to your directory classes. Ahh it is empty.

There is always a first:

Know for your first mod. All we are going to do is mimic what the fatboy mutator does. All this is showing is just the way that you would go on to creating more mutators etc. Go to your UnrealTournament->Botpack->classes folder. Open up fatboy.uc. After you have done this copy all the data on to the clipboard. Now goto your directory mymod->classes and create a new file called first.uc. After you have done that open the file with notepad, after this had been done then paste all the data. Here comes the tricky part. When UCC compiles your .UC files it is really nit-picky about class and file names The very first line of actual code in the fatboy.uc is

```
class FatBoy expands Mutator;
```

As you notice the class “fatboy” is the same as the name of the files fatboy.uc. This is what I was talking about when I said nit picking. When you code .uc files you have to make sure that your class names our the same as the actual filename. So in your case you would change that line to

```
Class first expands Mutator;
```

Well go back to your system directory; see the thing is every time you fire UCC you create .u files which the game reads. Well when you are creating your mod you are going to want to get rid of mymod.u because you keep changing it. The only .u files which you will delete are the ones which you have created, the ones like Botpack.u fire.u etc, you will not delete because you won’t change these. So in recap everytime you recompile your mymod make sure to go back and delete mymod.u.
Alright back to the system directory in the dos prompt and type ucc make.
You should get the same log. Now before you go and fire up UT you have to remember that you have to tell the unreal engine that you just added something and that you want the engine to run it. Well remember that mymod.int file you made that is the way you communicate to the engine. This is what you are going to want to add to it.

```
[Public]
Object=(Name=mymod.first, Class=Class,MetaClass=Engine.Mutator,Description="FatBoy clone.")
```

Well know that you have done this fire up Unreal Tournament and then go to start practice session go over to mutators and there you should see your “first” in the windows, if you look at the bottom of the screen you will see it’s definition called fatboy clone.

Conclusion:

Hell NO! This is just the beginning. AS a I stated earlier this is a work in progress so be sure to check back later.

Hints And Tips:

Well below you are going to find some hints and tips that I found very helpful for my self when I was coding UnrealScript. These are all tips that you guys will find useful since I did find these out on my own when coding UnrealScript.

Hint: You will use all these tips when you start coding and you will encounter all of these problems, without a doubt!

- PATCHES OVERWRITE THE CHANGES YOU MAKE TO YOUR UNREALTOURNAMENT.INI
- Every time you apply a new patch you have to redo your old changes with the packages (EditPackages) so that UCC will compile your packages.
- Upon load of UnrealEd some times you may get a error saying your packages is corrupt or something like that. Don't worry this is a normal thing for UnrealEd and it usually means that your "MyMod" is not included in the UnrealTournament.ini make sure you go there and add your name of your mod package to EditPackage. See about steps for more details.
- packages. Like by saying "you mod" not found or some crap like that.
- Use UnrealEd only once; the only time you will use this is when ever you are going to use the "export all" feature. After you do this never go back to UnrealEd. You might want to check with UnrealEd by opening if you have your EditPackges file settings right. The reason I say this is because UnrealEd loads this file up and will check to see if everything is working like compile wise. It is a shortcut, but I recommend not using it.
- I made have mentioned this early but when UnrealTournament loads it scans through the /UnrealTournament/system folder for all the .int files. This is due to the fact to get all the primary setting started and ready for the game to start using. Mostly menu related stuff.
- Never touch the actual .uc files that UnrealEd dumps. Just create a subclass of these files and modify them from there.
- the botpack package is the where you will find about the
- game modes.
-
- *TournamentGameInfo (all of the subclasses have this info. universal)
- *DeathMatchPlus (normal deathmatch)
- *TeamGamePlus (contains code for team-deathmatch as well as general team management)
- when you are doing actual coding and have expanded a gametype(the ones above)
- in {default properties} when you specify the {"GameName= "x"}, the x will
- show up in practice selection window, scoreboard, and other places where
- the name is mentioned. As you can see this variable is important.
- When specifying files and places stuff is located to load in your .int files remember to be very specific and remember that you must subclass everything. Look through the the other .int files like BotPack.int.
- When you want to test your changes which you added to the game you are going to have to use the summon command. This is done by going to the console and typing '~'. Type 'iamtheone' in the console. What this does is enable the cheat mode where you can add stuff etc. After this type summon and the name of your package then a dot and the filename. Ex.
  summon mymod."your_uc_filefortheweapon".