back <- up -> next

UT Mod Tutorial - 1: New Game Type

by: Robert "RJ" Martin

- This is a tutorial designed to give someone a quick start at making a Mod in UnrealScript. Like other tutorials, we will start by having you manipulate existing UT content and testing it out. Unlike other tutorials, however, we will start you off with the intent of you finishing with a completely new game. A knowledge of OO programming is assumed and some Java is helpful.
- Conventions:
  - Orange Text - refers to something interface specific, e.g.- Menu->File->New
  - Green Text - refers to something system specific like files or paths, e.g.- C:\WINNT\
  - `Code` Text - refers to something you will find in a file, e.g. - `class Thesbian expands Actor { }`

What You Need

A copy of **Unreal Tournament** (If you want to work outside the lab)

Includes: **Runtime** Environment, **UnrealEd**, and **UnrealScript Compiler**

- Make sure to include components like UTEd and when installing
- Make sure you have the latest UT patch: 4.36 patch for UT

A copy of **WotGreal for editing UnrealScript**

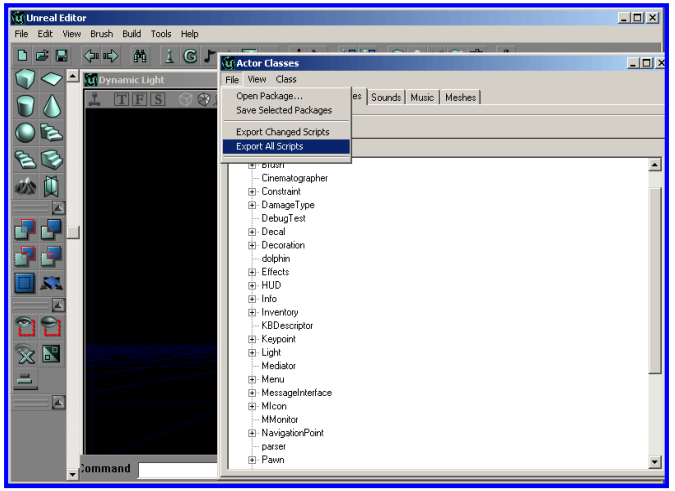
Available free of charge from wotgreal.com

- Extract WotGreal and install or wait until later in the tutorial

The first step is extract all of UT's source code using the UTEditor

- xxx.uc files are source files for UnrealScript. This step brings them from their source file format (xxx.u) that ships with UT back to source format for you to edit. You can do this with Mods you have downloaded and installed as well. You can think of the source exporter in UTEd as an application within the UT development environment that reverse engineers the content (but not the underlying engine) for you. You can extract models, textures, sounds, etc... but this step will only do the source code.

  - Open up UTEd
  - Open up the actor browser (the icon on the menu is a pawn) 
  - Goto Menu->File->Export All Scripts
  - You will see a dialog box that asks if you would like to create xxx.uc files for later use. Click Ok.
  - You should now see a list of Actor Classes in the Actor Browser and the main Actor classes from which they subclass. Feel free to expand parts of the tree to explore different types of Actors.
  - Close UTEd.



Now you must create the Package for your game

- A package is an organizational scheme used by UT to separate game content into discrete projects. Packages are represented by folders found directly under the UnrealTournament folder in the file hierarchy. Some examples of packages are: Engine, Core, and UnrealShare (If you do not see these under (C:\)UnrealTournament\, go back to the source code extraction step). What this means to you is that you can keep all of your code and content in your new package to keep it distinct from the content distributed with UT as well as that made by other content developers like yourself.

- Create a new folder in your UnrealTournament folder with whatever name you want. We will proceed using "myGame".
- Make a new folder in myGame called "Classes":
  (C:\)UnrealTournament\myGame\Classes
- This new folder is where your source files will go. The myGame folder is the package that your new game, source, media, etc... will be contained in and associated with. The folder name determines where the compiler will look for different file types.
  - \Classes - xxx.uc - UScript source files
  - \Textures - xxx.utx - texture files
  - \Models - xxx.3d, .psk, .psa, etc... model files which are converted at compile time
  - \Sounds - xxx.uax
  - \Music - xxx.umx
    - Models, Sounds, and Musics do not have to be in those respective paths, as you will load them or path them explicitly in most cases, but it is a good convention so we will stick to it.
- You can just create the Classes folder for now

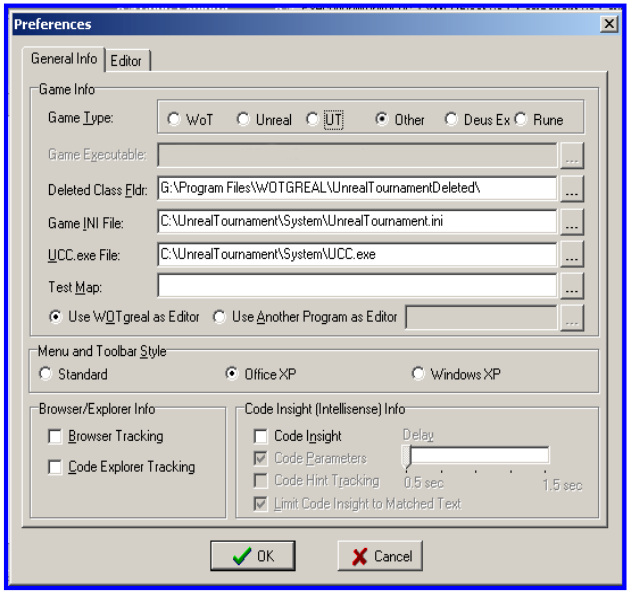
Editing UnrealScript code - 1: Configuring UT for your game

- Before you get started making content, you will have to configure UT so that it will recognize your new package and game type. Any Mod that you download had to do this at the beginning of its development. Fortunately, when it comes time for you to package your Mod, you will simply distribute your source, so the person playing doesn't have to do this step. However, you are essentially making your new Mod out of thin air, so you have to tell the UT engine that it is there and how to handle it during development. These are the first few steps in that process and should be enough to get you going.

Configuring WotGreal

It is not essential that you use WotGreal to edit your source. Any text editor will do. In fact, UTEd will also bring code up for you through the actor browser. However, WotGreal's interface is specifically designed for games that use the UT engine and all of the tutorials and reference material I have read advises AGAINST using UTEd to edit source at all costs. There are tutorials for using MSVStudio and such out there. I have had the most success with WotGreal, but it is a little buggy and idiomatic at times. The hardest part is getting it up and running the first time. After you have your new packages/games configured, it makes everything easier, but there is a little startup headache. If WotGreal freezes up during class/path refresh or startup, just close it and restart.

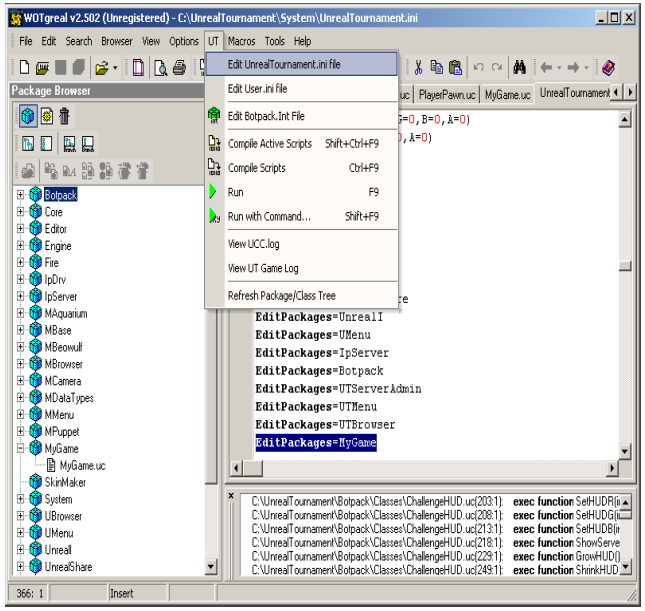
- If you have not run WotGreal before, you will be prompted to set up your new game's file paths immediately, otherwise go to the Menu->Options->Preferences
- Click the radio button that says, other or UT, whichever you prefer.
- The Game Executable section can be empty.
- the Deleted Class Path should be filled in by WotGreal. If not, just put it somewhere you will remember. This is where deleted xxx.uc files will be placed when you delete them through WotGreal's IDE.
- in the Game INI: text field, push the browse button (...) and open up the UnrealTournament.ini file. Start from wherever you have installed UnrealTournament:
  e.g.- (C:\)UnrealTournament\System\UnrealTournament.ini
- The next text field, UCC.exe:, should be filled in automatically for you when you specify the xxx.ini file
- ignore the text field, TestMap:
- choose OK/yes when prompted to expand the class tree, Menu->UT->refresh package/class tree



Configuring UnrealTournament.ini

xxx.ini files are used by the UT engine to store persistent data that is useful to the engine. xxx.ini files are usually associated with the engine in general and preferences, e.g.- User.ini stores player preferences; That's why they don't go away after you exit UT. In this case, we are saving changes to the compiler and UTEd configuration that were made since its release, namely your new Mod. Using WotGreal, we willl associate UnrealTournament with the myGame package we made before. Once we do this, the UnrealTournament compiler and game logic (ucc.exe contains both of these) will be able to reference it.

- From the menu, go to Menu->Tools->Modify/Edit Packages
- Add a package called "myGame". The package should now be associated with UT's compiler. Verify this by opening UnrealTournament.ini.
  - A quick way to open up the xxx.ini file associated with your game (UnrealTournament.ini at this point) is to go the the Menu->UT-> edit UnrealTournament.ini.
- Now do a search on the string "editpackages" (Ctrl+F, case insensitive). You should now see a highlighted portion of the file 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=myGame
```

If you do not see this last line (it will not be red), then add it yourself by hand and replace "myGame" with your package name.

- refresh the package tree, Menu->UT->refresh package/class tree

Editing UnrealScript code - 2: Creating Your New Game Type

Ok, we are finally ready to look at a little UnrealScript source code. If you don't know, UnrealScript is an OO language like Java or C++, so it supports inheritance. We DO NOT EVER edit classes we did not write ourselves. This will compromise the integrity of UT, plus we don't have access to native source to ever get it back to the way it was. If you accidentally change code and don't know what you did or delete a UScript object file (xxx.u), just get xxx.u files from someone else's distro or backup your package and re-install UT.

We make changes to the way things operate by subclassing and using xxx.ini and xxx.int files to tell the system which classes to use for different things. The class for our new game type will therefore be based on an existing game type, TournamentGameInfo. This will create a basic game with no rules, no HUD, no Bots, etc... You may certainly base your game type on DeathMatch, CTF, or whatever if you plan to focus more on changing things besides rules of the game. We use TournamentGameInfo because it provides multiplayer support and all the goodies that UT has to offer, but give you a clean slate to make a new game.

Subclassing From an Existing Game Type

From within WotGreal...

.

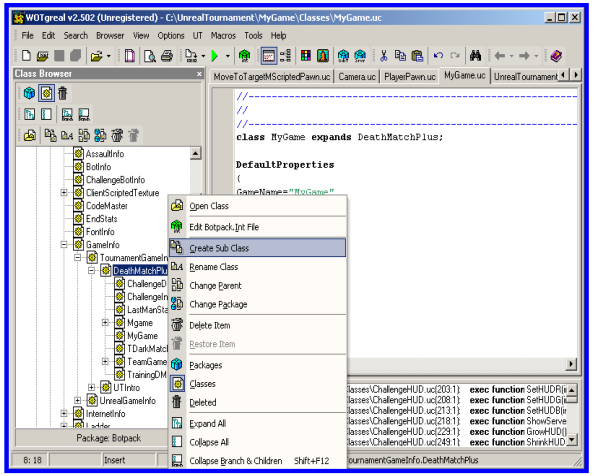
Now create a new text file called nothing.uc from within WotGreal using File->New. The file should have compilable content. Put this code in nothing.uc and save it in your packages (C:\)UnrealTournament\myGame\Classes. folder

```
class nothing expands object;

DefaultProperties
{
}
```

All that this does is allow WotGreal to access the package.

- Go to the Class Browser on the left  and expand the actor tree to find TournamentGameInfo.
  Object->Actor->Info->GameInfo->TournamentGameInfo.
- Right click on the TournamentGameInfo branch and choose Create SubClass. Choose myGame for your new class name and scroll through until you find myGame or type it in to the package name field.
- If prompted to create the package DO NOT say yes. This will make your package under the \System directory. This means that WotGreal cannot see your package.
- If WotGreal does not see your package, then go back and make sure that you have a (C:\)UnrealTournament\myGame\Classes folder with a compilable xxx.uc in it. WotGreal will look for any folders directly under UnrealTournament\ that, in turn, have a Classes\ folder directly under them. Only these packages will be usable in WotGreal.



Customizing Your New Game Type Through DefaultProperties

The DefaultProperties section is a very special section with a different parser, different syntax, and different behavior. It has links to the xxx.int files I will talk about shortly, but for now just think of it as a a very irritable constructor. That is to say, this is where you can put default startup values for your class.

You should now see the source file, myGame.uc. you can see that it extends TournamentGameInfo and has a small empty section called DefaultProperties.

- NOTE: do not type anything after the DefaultProperties section.
- In the DefaultProperties section type this line:

  GameName="MyGame"
- your file should now look like this:

```
class MyGame expands TournamentGameInfo;

DefaultProperties
{
GameName="MyGame"
}
```

- NOTE: This line must look exactly like this except for the part in quotes, no extra spaces or anything. UT is VERY particular about the syntax of the DefaultProperties section.

Configuring Your Game Type With xxx.int

Now we have to give a metaclass definition for our game type so UT will know in which package to look for its superclass. We will do this by creating an xxx.int file to be associated with our package. xxx.int files are similar to xxx.ini files, in that they are used for persistent data, except xxx.int files are associated with a certain package. As a result, we can use a file for instructions to the engine on how to load our game type without mucking with the UnrealTournament.ini on the user's system once we have finished our Mod and want to install it on their system. You will make changes to this file later as you progress, but this should be all you need for now. Just remember that this file is here.

right click on the myGame class branch in the class or package browser:  or  and choose edit myGame.int.

Add the following line:

```
[Public]
Object=(Name=MyGame.MyGame,Class=Class,MetaClass=Botpack.TournamentGameInfo)
```

Save the file, myGame.uc and minimize WotGreal

Compiling Your New Game Type

UTEd and WotGreal are pretty buggy, but you're in luck with the UCC compiler. It...Is...Rock...Solid... If there is anything you can rely on, it is this little piece of art...nay, miracle. The error messages are usually pretty helpful too, once you get used to UT syntax, anyway

- if this is your first time compiling, skip to the next (ucc make) step, otherwise you must delete the object file (xxx.u) associated with your package. The UT compiler will only compile packages that are listed in the EditPackages section of your game's xxx.ini file, and do not have a corresponding object file. Also, you can only reference classes that are defined in the same package or those compiled prior to yours.
  - Type the following line
    >del myGame.u
- open a command prompt (cmd or command) and change to the UT's system directory: (C:\)UnrealTournament\System\
  - type the following line:
    >ucc make
- If everything goes smoothly, you should see your package and class compile. Otherwise, you will have errors.

Testing Your New Game Type

- Start UT and from the menu, choose Multiplayer->Start New MultiPlayerGame. A dialog box will appear which shows the list of game types. Click on the listbox and choose myGame.
- A default map should start with your new game type being run within it.