

Creating, Modifying, and Compiling Classes

By [Jered](#)

Date Created: 5/23/2002

What Are Classes?

Classes are similar to objects. When you open ExEd, on the right side there is tree, with tons of information on it. These are weapons, lights, navigation points and so forth. Each one of those things has its own script, written in .uc format which has been compiled. To write code that can be compiled, you need a copy of Notepad, or another simple text editor. Classes are very complex, but easy to learn the basics of. A knowledge of UnrealScript (UScript) would be handy, but it is not required in simple class creation.

The basic theory of classes is a tree. It starts with a base that everything is connected too. The base you will be mostly working with is the 'Actor' class. Now, along with a tree, whatever is in the base is in the branches. So whatever you put in the 'Actor' class will be in all of the other classes coming off of that as well. What makes these other classes unique is what other stuff is in them. For example, the 'Info' class and the 'Pawn' class still have the same base, just they have other things added and modified to make them unique. Kind of like inheritance.

Once you have your base in, classes become unique, such as weapons, or inventory items. This is where you begin setting things you created with classes higher up. (higher up means closer to the 'Actor' class in this tutorial.) One of the basic types of things to set is the Boolean toggle, which is just a simple 'True' or 'False'.

Classes can be modified indirectly through ExEd. You can do this by placing the desired object in your level and then editing the properties. But this is highly ineffective for creating new objects en masse and impossible with creating modifications. What a class does, in a sense, is create a template.

Creating Your Own Class

In my opinion, the easiest class to create in is the 'DeusExWeapon' class. This is because it involves lots of simple strings and Boolean toggles. First off, open up notepad, and create a blank file. Now, one of the most basic, important, and useful things in creating classes is the comment. A comment is commentary for anyone who opens your file. it is a simple double slash "//". What this double slash does, is tells the compiler not to compile the rest of the line. An example of how it works:

```
//This is not being read by the compiler  
//I can put anything I want here without messing up the class.
```

The comment is a great tool. It is extremely helpful to others using your code and yourself. For instance if your going back through your code and you say "Why the heck did I do this?" the comments will help you figure it out. I cannot stress enough how important it is to have well commented code.

Now, it has been the general rule that when creating classes that you uses a special header in a certain format. This denotes what the class is and makes it look cool. It is also the

format used by Epic coders and ION Storm coders alike. It is:

```
//=====
//[name of your class].
//=====
```

Like I said this is optional, but it helps to state what it is when modifying it. The next thing that is necessary when creating class is to define where the new class will go. You need to include this simple line:

```
class [Name Of New Class] extends [Class It Comes Off Of];
```

This designates where its going. For example, if I made a class that contained the line

```
class People extends Pawns;
```

it would create a new class under pawns called People. Another thing with this, is that the class you are creating should have the same name as the title of the .uc file.

Those are the two most basic things in creating your own class. The next thing is the base of class. It is the 'defaultproperties'. This, like all other things, is case sensitive. The format for the default properties command is:

```
defaultproperties
{
[What the Default Properties are]
}
```

It must be enclosed in brackets. Setting default properties is very simple. To find out what you can set, create an object of the class your modifying in your level. Then open up its properties. Notice how some things are always set to a certain property. This is what 'defaultproperties' does. Now, in these properties, open up a tree and expand it all the way. Notice at the bottom of the tree, it goes to just a bunch of fields you can set. next to these blank fields there are the names of the fields.

The left side is the names of the fields, and the right is the settings for that field. It works similar in the .uc file. You just put:

```
defaultproperties
{
[name of field]=[the settings for that field]
}
```

Its that simple. Note that all these to are case sensitive. You now have enough knowledge to create your own class. Open up notepad. Put in your intaductory commentary, which looks like this:

```
//=====
//WeaponStunRifle.
```

//=====

then you need to put in your class location definition.

```
class WeaponStunRifle extends DeusExWeapon;
```

This sets it up to go under 'Actor -> Inventory-> Weapon -> DeusExWeapon' Now that that is all set, you can begin setting its default properties.

```
defaultproperties  
{  
BaseAccuracy=0.1  
bAutomatic=True  
bHasLaser=True  
bHasSilencer=True  
bHasScope=True  
HitDamage=1  
recoilStrength=0  
reloadTime=4  
ScopeFOV=5  
PickUpAmmoCount=12  
ReloadCount=1  
StunDuration=15  
Name=WeaponRifle1  
ItemName="Stun Rifle"
```

Description="The stun rifle modification was developed in order to increase accuracy of the modern rifle. With the modification, the rifles base accuracy is 90%. An added side effect is that since the bullet travels through the tube, it gets charged with electricity. The Electricity in the bullet stuns the victim for a long amount of time. The downside to this is that the reload rate is low and only one bullet can be in the chamber at the time. In addition to creating the electric charge, the bullets also are slowed significantly, lowering the amount of damage inflicted."

```
beltDescription="STUN RIFLE"  
Mesh=LodMesh'DeusExItems.SniperRiflePickup'  
CollisionRadius=26.000000  
CollisionHeight=2.000000  
Mass=30.000000  
InventoryGroup=60  
PlayerViewOffset=(X=20.000000,Y=-2.000000,Z=-30.000000)  
PlayerViewMesh=LodMesh'DeusExItems.SniperRifle'  
PickupViewMesh=LodMesh'DeusExItems.SniperRiflePickup'  
ThirdPersonMesh=LodMesh'DeusExItems.SniperRifle3rd'  
LandSound=Sound'DeusExSounds.Generic.DropMediumWeapon'  
Icon=Texture'DeusExUI.Icons.BeltIconRifle'  
largeIcon=Texture'DeusExUI.Icons.LargeIconRifle'  
largeIconWidth=159  
largeIconHeight=47  
invSlotsX=4
```

```

bInstantHit=True
FireOffset=(X=-20.000000,Y=2.000000,Z=30.000000)
shakemag=50.000000
FireSound=Sound'DeusExSounds.Weapons.RifleFire'
AltFireSound=Sound'DeusExSounds.Weapons.RifleReloadEnd'
CockingSound=Sound'DeusExSounds.Weapons.RifleReload'
SelectSound=Sound'DeusExSounds.Weapons.RifleSelect'
bUseWhileCrouched=False
bCanHaveModBaseAccuracy=True
bCanHaveModReloadCount=True
bCanHaveModAccurateRange=True
bCanHaveModReloadTime=True
bCanHaveModRecoilStrength=True
AmmoName=Class'DeusEx.Ammo3006'
bCanHaveLaser=True
bCanHaveSilencer=True
bCanHaveScope=True
GoverningSkill=Class'DeusEx.SkillWeaponRifle'
NoiseLevel=2.000000
EnviroEffective=ENVEFF_Air
ShotTime=1.500000
}

```

I definitely suggest cut and paste for that. If you will notice, all of these statements follow the basic formula. Now, what this will do, after its compiled, is create a sniper rifle with 90% accuracy, a scope, silencer, a one shot chamber, and a laser. But the bullets will stun the victim instead of killing him,. Now on too compiling.

Things To Remember:

Comments, there are three basic types that do basically the same thing

The beginning of UC file header:

```

//=====
//[UC File Name].
//=====
The slightly lower middle of file comment
//-----
//[What Area This is]
//-----

```

And the quick note:

```

// [Note]

```

Another thing to remember is white space. As with C++, spacing does not matter. If it helps you organize the file, put commands 50 spaces apart. The only place you cannot have white space is in the middle of a phrase

Acceptable:

```
class WeaponStunRifle extends DeusExWeapon;
```

Unacceptable:

```
class WeaponStunRifle ext ends DeusExWeapon;
```

Another thing is the way fields are done. If you are setting something like the 'Description' field which requires custom text, you need to enclose it in quotes.

```
ItemName="Stun Rifle"
```

Compiling Your UC File

UC stands for Unreal Compile. In order to get your file into a format Deus Ex understands, you need to compile it. The compiler for UC files is included with the DeusEx SDK. Its called UCC.exe. Now, like most compilers, it requires that everything be in a certain place. The first major thing is the actual UC files location. Create a directory in your DeusEx directory called 'WeaponStunRifle'. Then, in that directory, create a folder called 'Classes'.

When you run ucc, it only compiles the uc files in the classes folder. Again, typing it right is essential. Now, you need to place your uc file in the 'Classes' directory. Make sure the file is named "WeaponStunRifle" and that it has a file extension of '.uc'. To change the file extension rename the file "WeaponStunRifle.uc" Now the second thing is editing your DeusEx.ini. This is where things get a bit tricky. Open it up and search for the section that has all the 'EditPackages=' lines. Now, add the line:

```
EditPackages=WeaponStunRifle
```

The part after the '='s must be the name of the folder or else it won't work. Now, we have to compile the actual thing itself. Go to an MD-DOS prompt. Type, in order with an enter after each line,

```
cd\;  
cd\DeusEx\system  
ucc make
```

If it went well, when it goes over the package DeusExWeaponRifle, it should parse and build it. Now, all that is left is to restart ExEd and there you have it! A new class! Note that for recompiling .u files, you must delete them. Then re-run ucc make. If it gives you any errors, go back and check over what you put in the UC file. Well that is about it for Creating and Compiling your own class.

"Give us the Tools, and we will build worlds" ~ Jered