UT Model Importing from Milkshape 3d

The lastest version of Milkshape 3D can be found here:

http://www.swissquake.ch/chumbalum-soft/

Purpose:

This tutorial is specific to Milkshape 3d and is to instruct those who wish to use Milkshape to make models and then import them into Unreal Tournament. This tutorial describes only one way to do it, but a way in which I am familiar with and have found to work well. It will not however teach on the methods of making or animating of a model, as that is another tutorial alltogether.

Glossary:

Mesh - A mesh simply a 3D model.(usually without a texture)

Texture - A 2D image. This can refer to textures external to UnrealEd, such as .pcx or .bmp files, or to textures imported into one of your packages.

* The height and width of your textures need to be "powers of 2." In other words, both the height and the width need to be one of these numbers: 1, 2, 4, 8, 16, 32, 64, 128, or 256. You can go higher (512, 1024, etc), but that doesn't seem to be typical when using the Unreal engine (yet). The height and width do not need to be the same, so a 64 x 32 image is fine, even a 256 x 2 is allowed if you need it.

Vertex - A single point in 3D space. These are comprised of 3 infomational factors: the X, Y, and Z co-ordinates.

Polygon - In the Unreal engine, this describes three vertices (points) that are "connected" in a particular order to make triangles. These triangles form the solid part of your model (the part that you actually see). A triangle is the simplest type of polygon. In Milkshape these are considered as "faces".

Surface - A group of polygons in your mesh that can be referenced as one and can have a set of UV texture mapping coordinates associated with them. In MilkShape, each Group that you create will end up being a surface.

UV Coordinates (a.k.a. Texture Mapping Coordinates) - A set of 2D coordinates that refer to a row and column position on a texture image. In the Unreal engine, every polygon is a triangle made up of three 3D points. Every polygon in a surface can also be associated with three 2D points that refer to a triangle-shaped region of a texture. These 2D points are called UV coordinates. The "U" refers to the column position and the "V" refers to the row position. It's called UV instead of XY so that it's not confused with X, Y, and Z when talking about 3D coordinates.

1. Create a kickass model

Once you have successfully created your mesh, including texturing and animation, you're ready to go on. If you want to create a generic mesh that can be used multiple times with different textures (skins), you'll still need to assign texture coordinates (UV coordinates) to your mesh using reference textures.

2. Export your mesh to Unreal format

In order to get your model into unreal you must first set up for it. This entails creating a package which we will talk about later.

First off, you need to set up your folders so unreal will know where to look when we do make our package. This is done as so:

Under your UnrealTournament folder, create a new folder with the name of your package(or what it will be). Then, under your package folder place new folders with the following names: Classes, Models, Textures, and optionally Sounds(if you want to create and import .wav files for new sounds).

Your next step will be to export your model from Milkshape in the UT format and save them in your package's Models folder.

Upon export from milkshape you'll end up with a pair of files ending with .3d. One will end with _a.3d and the other with _d.3d. You will also get a pre-generated .uc file which is pretty generic. We'll get into that later, but you'll probably end up needing to edit at least one .uc file.

You'll also want to convert all of your textures to .pcx or. bmp format if necessary. Note that only 256-color (8-bit) textures are supported by the Unreal engine. This isn't as bad as it may sound, since each image can have its own palette.

3. Create or modify a .uc file for importing the mesh into Unreal.

This can be created or edited in any text editor(I simply use notepad, there are many other softwares available to do this). Move or save this .uc file in your Classes folder that is under your package's folder. This is what it should look like.

```
class testGun expands Actor;

#exec MESH IMPORT MESH=myGun ANIVFILE=MODELS\gun1_a.3d DATAFILE=MODELS\gun1_d.3d X=0 Y=0 Z=0
#exec MESH ORIGIN MESH=myGun X=0 Y=0 Z=0

#exec MESH SEQUENCE MESH=myGun SEQ=All STARTFRAME=0 NUMFRAMES=25
#exec MESH SEQUENCE MESH=myGun SEQ=Still STARTFRAME=0 NUMFRAMES=1
#exec MESH SEQUENCE MESH=myGun SEQ=Fire STARTFRAME=1 NUMFRAMES=20
#exec MESH SEQUENCE MESH=myGun SEQ=Reload STARTFRAME=21 NUMFRAMES=5

#exec MESHMAP NEW MESHMAP=myGun MESH=myGun
#exec MESHMAP SCALE MESHMAP=myGun X=1 Y=1 Z=.5

#exec TEXTURE IMPORT NAME=myGunTex1 FILE=TEXTURES\guntex1.pcx GROUP=Skins FLAGS=2
#exec TEXTURE IMPORT NAME=myGunTex1 FILE=TEXTURES\guntex1.pcx GROUP=Skins PALETTE=myGunTex1


#exec TEXTURE IMPORT NAME=myGunTex2 FILE=TEXTURES\guntex2.pcx GROUP=Skins FLAGS=2
#exec TEXTURE IMPORT NAME=myGunTex2 FILE=TEXTURES\guntex2.pcx GROUP=Skins PALETTE=myGunTex1

#exec MESHMAP SETTEXTURE MESHMAP=myGun NUM=1 TEXTURE=myGunTex1
#exec MESHMAP SETTEXTURE MESHMAP=myGun NUM=2 TEXTURE=myGunTex2
#exec MESHMAP SETTEXTURE MESHMAP=myGun NUM=3 TEXTURE=myGunTex1

defaultproperties
{
DrawType=DT_Mesh
Mesh=myGun
}
```

To understand everything that's going on here, let's break down all of the types of statements you'll need to have in your file.

Class Definition Statement

```
class testGun expands Actor;
```

Well, right off the bat we've got a problem. When milkshape produced your .uc file it didn't know anything about how you plan on using your mesh, so it just set it up as a generic "Actor". This is rarely (if ever) what you'll want when you get to the point of really using your mesh for something (like for a weapon or character). Even if it's just a "decoration", you'll want to change this. We'll get back to this later, but for now this is enough to get the mesh into Unreal. It will show up in the UnrealEd Mesh Viewer and as something you can place in a map. You just won't be able to do much with it.

For now, replace "myGun" with what you want to call your new object. Do not include any spaces.(This new object name must also match the name of the .uc file or you will get an error when compiling)

Mesh Import Statement

```
#exec MESH IMPORT MESH=myGun ANIVFILE=MODELS\gun1_a.3d DATAFILE=MODELS\gun1_d.3d X=0 Y=0 Z=0
```

This one's easy. Just replace the _a.3d and _d.3d names with the names of your .3d files and call your mesh something after the "MESH=". This tells the Unreal compiler where to find your .3d files. In this example, the .3d files would be stored in a Models folder under your package folder.

Origin Statement

```
#exec MESH ORIGIN MESH=myGun X=0 Y=0 Z=0
```

For most items like weapons or pickup items, you'll want to just leave this statement as is, except for changing the mesh name. The MESH ORIGIN statement sets up where the center of the mesh is. In other words, the point where your mesh will rotate around and move from.

You can also use the MESH ORIGIN statement to set the initial rotation values of your mesh. If you've modeled a weapon and it ends up facing the wrong direction in the game, you can fix that here. You do that by adding PITCH, ROLL, and/or YAW values to the statement. The values aren't angles though as the unreal engine relies on a system using of multiples of 16 units. A value of 64 means 90 degrees, a value of 128 means 180 degrees, 256 means 360 degrees, etc. So this statement:

```
#exec MESH ORIGIN MESH=myGun X=0 Y=0 Z=0 PITCH=32 ROLL=128 YAW=-64
```

...would mean a pitch of 45 degrees, a roll of 180 degrees, and a yaw of -90 degrees.

Animation Sequence Statements

```
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=All  STARTFRAME=0  NUMFRAMES=25
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=Still  STARTFRAME=0  NUMFRAMES=1
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=Fire  STARTFRAME=1  NUMFRAMES=20  RATE=24
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=Reload  STARTFRAME=21  NUMFRAMES=5  RATE=7.5
```

If your mesh includes animation, you'll usually need to name your various animation sequences in these statements. When Milkshape generates .uc files when you create your .3d files , it doesn't do much for you here. By convention, the first sequence name is called "All" and simply includes all of your animation frames. Create a MESH SEQUENCE statement for each of your animation sequences and give each a meaningful name. Unless the sequence is only 1 frame long, you may want to add a "RATE=" statement to the end of the line. This sets up how long it will take to play the sequence, in frames per second. Milkshape by default uses 24 fps so a RATE of 24 will set animations to the same speed witnessed when they were created.

If your mesh does not include any animation, you can just use something like this(which is good for weapon pickup and third person view meshes):

```
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=All  STARTFRAME=0  NUMFRAMES=1
#exec  MESH  SEQUENCE  MESH=myGun  SEQ=Still  STARTFRAME=0  NUMFRAMES=1
```

Meshmap to Mesh Associations

```
#exec  MESHMAP  NEW  MESHMAP=myGun  MESH=myGun
```

Don't worry about this one; just replace myGun with the name of your mesh and you'll be fine.

Meshmap Scaling Statement

```
#exec  MESHMAP  SCALE  MESHMAP=myGun  X=1  Y=1  Z=.5
```

Scaling is an important concept to understand. The way that Milkshape's "units" relate to Unreal units is easy: 1 MilkShape unit will become 1 Unreal unit. That way if you treat 16 units in MilkShape as 1 foot, you should be able to set your scaling to "X=1 Y=1 Z=.5" The traditional Unreal format works like this: Z will be half of what X and Y are. Not really sure why.

Texture and Palette Import Statements

```
#exec  TEXTURE  IMPORT  NAME=myGunTex1  FILE=TEXTURES\guntex1.pcx  GROUP=Skins  FLAGS=2
#exec  TEXTURE  IMPORT  NAME=myGunTex1  FILE=TEXTURES\guntex1.pcx  GROUP=Skins  PALETTE=myGunTex1
```

For every new texture image that your mesh uses, you'll need a line to import it. Give it a meaningful name in place of "myGunTex1" and fill in the name of the .pcx file. In this example, the .pcx files would be in the \Textures folder under your package folder. You can change that to something else if you want, but this is the most common format.

Meshmap to Texture Associations (optional)

```
#exec  MESHMAP  SETTEXTURE  MESHMAP=myGun  NUM=1  TEXTURE=myGunTex1
#exec  MESHMAP  SETTEXTURE  MESHMAP=myGun  NUM=2  TEXTURE=myGunTex2
#exec  MESHMAP  SETTEXTURE  MESHMAP=myGun  NUM=3  TEXTURE=myGunTex1
```

You'll almost always want to include these types of statements, unless you're creating a generic mesh. These statements associate the various surfaces in your mesh with specific textures. In MilkShape, you will get one of these for each Group you create. Each Group will be a "surface" that is numbered 1 to whatever. The "NUM=1" line says to associate the first surface with the myGunTex1 texture. Note that you can refer to the same texture from multiple surfaces. In this case, you'd only need to import myGunTex1 once, though.
Since the maximum number of surfaces allowed is 8, you would never have more than 8 of these statments per mesh.

Default Properties

```
defaultproperties
{
DrawType=DT_Mesh
Mesh=myGun
}
```

Drawtype simply tells unreal that you want this to be a mesh. All models will most likely be DT_Mesh.

Remember to save your .uc file in the \Classes folder under your package folder and name it the same as the class. The example would have been named testGun.uc.

5. Optional: add to .uc file the code that tells UT what you want your model to actually "do" or "be" ingame.

Your mesh is just a 3D object with no interesting properties or behavior. You'll typically want to use your mesh as a weapon, character, pickup item, or at least something you can interact with in some way. The way to do that is by scripting your .uc file so that it "expands" on an existing type of object, such as Weapon (for weapons), Pickup (for things that can be picked up and put into your inventory), or Decoration (for stuff like furniture). Each type of object that you can create (there are many), has a set of properties that you can set. One of the properties is which mesh or meshes to use. Some objects may have multiple meshes associated with them. For instance, a weapon has a "first person" version of the mesh that includes the player's hand and part of his arm, as well as a "pickup" version that is used to show that weapon sitting on the ground and a "third person" view which is what other players will see you carrying.

Without going into a lot of detail, here's a short example to give you the idea. The part in quotes after LodMesh would be the reference to your

mesh (that could have been imported seperately in another .uc file).

```
class weedplant extends Decoration;

defaultproperties
{
bCanBeBase=True
ItemName="Cannabis Sativa"
Mesh=LodMesh'MyPackage.weedplant'
CollisionRadius=18.790000
CollisionHeight=6.750000
Mass=10.000000
Buoyancy=15.000000
}
```

6. Import it into the game!

When you have your .uc files, your .3d model files, and your .pcx texture files in place, you're ready to actually import your mesh into your package, so that it's ready to use in Unreal. If you've ever compiled a package with "ucc make" then you already know how to do this step. Delete your .u package file(if it already exists) from the \UnrealTournament\System folder and in a command prompt (MS-DOS prompt) in the \UnrealTournament\System directory type in "ucc make".(or under your start menu, choose "run" and type "C:\UnrealTournament\System\ucc make"). If you've done everything right, and you recieve no errors or warnings, your mesh is ready to go. Fire up UnrealEd and enjoy.

Oh yeah, you may need to delete your .u package file and adjust the alignment in your .uc file then re-compile quite a few times to get it right so don't be discouraged, it will come to you.
(To align your weapon's first person view, just add the line PlayerViewOffset in default properties and give a value for each of the X, Y, and Z co-ordinates. Z being up and down in unreal. And remember, playerviewoffset manipulates YOUR PERSPECTIVE of the weapon, NOT THE ACTUAL WEAPON!)