Unreal Model & Animation File Format

Author: Bob Berry
Contact: bob@vsmm.org
Last Modified: 2/14/99

Overview

While you can import .DXF brushes directly into your map in UnrealED for construction, you are unable to dynamically spawn such objects in your world. In addition to that, the .DXF format does not contain animation information. The solution to creating reusable animated models for your worlds is to compile them into your packages using Unreal's proprietary data format.

The only available converters at the time of this writing are a 3D Studio Max to Unreal converter, called 3DS2UNR, developed by Legend Entertainment and a Quake2 model viewer that exports to Unreal format called Crusher md2 Model Viewer, developed by Tom Condor. The purpose of this document is to describe the file formats in detail for others who wish to develop a converter.

Files

Unreal stores the geometry and animation data in two separate files. These files are interdependent of each other, as Unreal needs to read both of them at the same time to construct an accurate mesh. The file names for these should be in the format of:

_d.3d : Geometry Data
_a.3d : Frame (animation) Data

Each file contains a block of header data followed by a series of geometry or frame data blocks. The geometry data format stores information in triangles with numbered unique vertices. The animation data format stores the position of the vertices at each frame. I will refer to the numbered unique vertices stored in the geometry data file as the "Vertex Map" to avoid confusion with the vertex positions.

The blocks labeled "Unused" are read by the importer, but are not referenced anywhere in Unreal. It is safe to zero them.

Data Types

The following is a list of useful data types and sizes as defined by Unreal.

`unsigned char`	`BYTE`	8-bit unsigned.
`unsigned short`	`_WORD`	16-bit unsigned.
`unsigned int`	`DWORD`	32-bit unsigned.
`signed int`	`INT`	32-bit signed.
`signed int`	`UBOOL`	Boolean 0 (false) or 1
`float`	`FLOAT`	32-bit IEEE floating point.

THE _D.3D FILE

The overall format of the geometry data file is as such:

[Geometry Header]
[Triangle Description 1]
[Triangle Description 2]
[...]
[Triangle Description N]

Geometry Header - Begins at: `0x00h` (36 bytes long)

`0x00h` Number of Polygons (Triangles) - 2 bytes _WORD
The total number of triangles in the mesh

`0x02h` Number of Vertices - 2 bytes _WORD
The total number of unique vertices

`0x04h` Bogus Rotation - 2 bytes _WORD
Unused

`0x06h` Bogus Frame - 2 bytes _WORD
Unused

`0x08h` Bogus Normal X - 4 bytes DWORD
Unused - zero out

`0x0Ch` Bogus Normal Y - 4 bytes DWORD
Unused - zero out

`0x10h` Bogus Normal Z - 4 bytes DWORD
Unused - zero out

`0x14h` Fix Scale - 4 bytes DWORD
Unused - zero out

`0x18h` Unused 1 - 4 bytes DWORD
Unused - zero out

`0x1Ch` Unused 2 - 4 bytes DWORD
Unused - zero out

`0x20h` Unused 3 - 4 bytes DWORD
Unused - zero out

`0x24h` Unused - 12 bytes -
Unused - zero out
This block is not even read by UnrealED during import

Triangle Definition - Begins at: `0x30h` (16 bytes per triangle)

`0x30h` Vertex 1 - 2 bytes _WORD
The first unique vertex this triangle is made of

`0x32h` Vertex 2 - 2 bytes _WORD
The second unique vertex this triangle is made of

`0x34h` Vertex 3 - 2 bytes _WORD

The third unique vertex this triangle is made of

0x36h Triangle Type - 1 byte BYTE
Bit vector of triangle type
(See Appendix A: Triangle Types)

0x37h Color - 1 byte BYTE
Color for flat and gouraud shading

0x38h Texture Triangle Coordinates 1, U - 1 byte BYTE
0x39h Texture Triangle Coordinates 1, V - 1 byte BYTE
First (u,v) texture coordinate for triangle
(See Appendix B: Texture Triangles)

0x3Ah Texture Triangle Coordinates 2, U - 1 byte BYTE
0x3Bh Texture Triangle Coordinates 2, V - 1 byte BYTE
Second (u,v) texture coordinate for triangle
(See Appendix B: Texturing)

0x3Ch Texture Triangle Coordinates 3, U - 1 byte BYTE
0x3Dh Texture Triangle Coordinates 3, V - 1 byte BYTE
Third (u,v) texture coordinate for triangle
(See Appendix B: Texturing)

0x3Eh Texture Number - 1 byte BYTE
Offset to textures imported by #execs in the .uc file
(See Appendix B: Texturing)

0x3Fh Flags - 1 byte BYTE
Unused - zero out

[Repeat Triangles as necessary]

THE _A.3D FILE

The overall format of the frame data file is as such:

[Frame Header]

[Frame Data Set 1]
[Frame Data Set 2]
[...]
[Frame Data Set N]

Frame Header - Begins at: 0x00h (4 bytes long)

0x00h Number of frames - 2 bytes _WORD
Total number of animation frames (1 for non-animated models)

0x02h FrameSize - 2 bytes _WORD

Frame Data - Begins at: 0x04h (4 bytes * Vertices per frame)

0x04h First vertex position of first frame - 4 bytes INT

Packed (x,y,z) coordinate of first unique vertex
(See Appendix C: Vertices and Appendix D: Frames)

`0xnnh` Last unique vertex of first frame

[Repeat vertex list for new frames as necessary]

Appendix A: Triangle Types

The type of the triangle is expressed as a bit vector. However, the first few bits define the exact nature of the triangle and therefore only one type must be used. The remaining bits are extra effects and may be added at will.

Triangles may have ONE of the following:

Normal one-sided	**0**
Normal but two-sided	**1**
Translucent two-sided	**2**
Masked two-sided	**3**
Modulation blended two-sided	**4**
Weapon triangle	**8**

Add any combination of the following numbers for extra effects:

Full brightness, no lighting	**16**
Flat surface, don't do bMeshCurvy	**32**
Environment mapped	**64**
No bilinear filtering on this poly's texture	**128**

E.g. 84 (2 + 16 + 64) = Translucent two-sided, full brightness, environment mapped polygon. Now wouldn't that be interesting....

Appendix B: Texturing

Each triangle has a triangular portion of a texture mapped to it. This process is called 'skinning'. The three texture coordinates specified for every triangle defines the triangular segment of the texture to copy and paste to the triangle.

To import the model in unreal you must also import a texture, otherwise your model will show up invisible in the mesh viewer. Each model can have any number of textures imported along with it. This is done in the .uc file that you must create to direct Unreal to compile the model into your new package. Here's an example of the directives needed to import your model and textures:

```
#exec MESH IMPORT MESH=test ANIVFILE=wee_a.3d DATAFILE=wee_d.3d X=0 Y=0 Z=0
```
Specifies the frame data file (ANIVFILE) and geometry data file (DATAFILE)

```
#exec MESH ORIGIN MESH=test X=0 Y=0 Z=0
```
Use this for uncentered models

```
#exec MESH SEQUENCE MESH=test SEQ=All STARTFRAME=0 NUMFRAMES=20
#exec MESH SEQUENCE MESH=test SEQ=Idle STARTFRAME=1 NUMFRAMES=5
#exec MESH SEQUENCE MESH=test SEQ=Walk STARTFRAME=6 NUMFRAMES=14
```

Animation sequences, specifies the name of the sequence, the beginning frame and how many frames it is.

```
#exec MESHMAP NEW MESHMAP=test MESH=test
```
Creates a new meshmap that corresponds to a mesh.

```
#exec MESHMAP SCALE MESHMAP=test X=0.1 Y=0.1 Z=0.2
```
Scales the meshmap

```
#exec TEXTURE IMPORT NAME=tex FILE=Textures\MyPix_01.PCX GROUP=Skins FLAGS=0
```
Imports a texture into a group called 'Skins', names the texture, and sets special flags (i.e. translucent)

```
#exec MESHMAP SETTEXTURE MESHMAP=test NUM=1 TEXTURE=tex
```
Assigns the texture to be used on the model.

The important thing here is the NUM=1 seen on the last line. That is the texture index number the triangle definition refers to.

Appendix C: Vertices

The packed vertex should be stored in a single 4-byte signed int. The XY and Z coordinates of the vertex are 4-byte floats and bit shifted for compression. Be careful of platform specific variations of type sizes.

Borrowed from 3ds2unr:
```
Packed Vertex = ( int( v.X * 8.0 ) & 0x7ff ) | ( ( int( v.Y * 8.0 ) & 0x7ff ) << 11 ) | ( ( int( v.Z * 4.0 ) & 0x3ff ) << 22 )
```

Appendix D: Frames

Each frame data block contains the list of unique vertex locations. As the animation progresses, the vertices move to different locations, however the vertex map specified in the geometry data format stays the same. The vertices must be listed in the same order for each from and each vertex must be listed, even if there has been no change in position. This maintains the integrity of the vertex map.

E.g. For two polygons:
Frame 1
Vertex 1 (12,15,23)
Vertex 2 (-15,23,-35)
Vertex 3 (100,-53,85)
Vertex 4 (41,15,21)
Vertex 5 (15,73,35)
Vertex 6 (78,-62,-23)
Frame 2
Vertex 1 (42,25,33)
Vertex 2 (65,33,35)
Vertex 3 (120,53,55)
Vertex 4 (45,-15,31)
Vertex 5 (122,-133,155)
Vertex 6 (-78,102,43)
The vertex map still reads
Triangle 1
1,2,3
Triangle 2
4,5,6