T3D File Format Specification

Mike "Mongo" Lambert (mongo@planetunreal.com)
Sponsored by Perilith Industrielle, developers of Unrealty

There are four types of objects that a T3D file can contain: polygons, brushes (groups of polygons), actors, and maps (groups of brushes and actors). Polygons and actors are never included on their own; they are only used to make up brushes and maps, respectively.

Polygons

Each polygon consists of multiple properties, and a list of vertices which define the polygon.

```
Begin Polygon [Item=<ItemName>] [Texture=<Texture>] [Flags=<Flags>] [Link=<Link>]
      [Origin     <BaseVector>]
      [Normal     <NormalVector>]
      [Pan        [U=<UPan>]          [V=<VPan>]]
      [TextureU   <TextureUVector>]
      [TextureV   <TextureVVector>]
      Vertex      <VertexVector>
      Vertex      <VertexVector>
      ...
End Polygon
```

<ItemName>
: **Not Used.** The name of the polygon. Only used when selecting polygons within the editor. Will be saved/reloaded upon export regarding the selected polygons of the brush

<Texture>
: The texture of the polygon. Must be specified in Package.Texture format.

<Flags>
: The flags of the polygon. The flags and their meanings can be found in the polygon flags section at the end of this document. It ignores the NoImport flag upon importing.

<Link>
: This is the index of the surface for this particular polygon. In general, there is one surface per polygon. The default for this is the number of the polygon, allowing each polygon to have it's own surface. In some cases, such as when "Merge Coplanars" is used on a brush, or a brush is validated, any coplanar polygons will have their iLinks set to point to the same surface, indicating that all of the polygons lie on the same plane.

<Origin>
: The origin of the polygon. If not specified, it defaults to the first Vertex listed.

<Normal>
: The normal of the polygon. Upon import, UT ignores and recalculates the normal itself. The normal is the vector heading towards you when you look at the polygon such that its points are listed in a counterclockwise fashion. In other words, the normal is the cross product of a line segment with it's subsequent line segment. This property (in the t3d file) is only useful upon export when one needs the normals without calculations.

<UPan> & <VPan>
: The U (X) and V (Y) offsets. These define the offset with which the texture is placed into the polygon, so that the texture does not need to be flush with the polygon. Setting UPan to five shifts the texture five units to the left, obscuring the leftmost five units of the texture.

<TextureUVector> & <TextureVVector>
: The U (X) and V (Y) directions on this polygon. Changing these causes the orientation as well as the scaling of the texture to vary. The TextureU and TextureV can be thought of as the vectors defining the coordinate system. Their direction defines the direction that the texture is applied to the polygon. The TextureU vector should be pointing to the right side of the polygon's texture, while TextureV should be pointing to the bottom of the polygon's texture. For normal scaling, these vectors should be unit vectors in their respective directions. Doubling either of the magnitudes of these vectors causes a doubling in the size of the texture in that respective direction. By default, TextureU is the normal parallel to the *first* line segment listed, while TextureV is perpendicular to that and the normal, a normal pointing downwards. Since they are both normals, they will cause the texture to be applied with a 1:1 scaling by default.

<VectorVector>
: This is a point on the polygon. These points should be listed in counterclockwise order around the polygon, so that the normal will be facing into the viewer.

Brushes (PolyLists)

Each brush is composed of various polygons connected together, forming a complete brush. The general format of such a brush is simple, as there is no information associated with it other than the polygons it contains.

```
Begin PolyList
      Begin Polygon
            ...
      End Polygon
      Begin Polygon
            ...
      End Polygon
      ...
End PolyList
```

Actors

The actor export lines store the specific properties associated with that actor, and how they differ from the defaultproperties as stored in the class itself.

```
Begin Actor Class=<ClassName> Name=<Name>
      <PropertyName>=<PropertyValue>
      <PropertyName>=<PropertyValue>
      ...
End Actor
```

<ClassName>
: The classname of this actor, in the format PackageName.ClassName

<Name>
: The unique identifying name of this actor

<PropertyName>
: Some property of the actor

<PropertyValue>
: The value of the associated property of the actor, as it differs from the actor's defaultproperties

One important point to mention is that brushes are also actors. So brushes can follow the same form, to a degree. They follow the standard "Begin Actor" format, but their contents change to allow for polygon list definitions, which give the information needed to reconstruct the brush. This is the form used specifically with map files.

```
Begin Actor Class=Brush Name=<Name>

      Begin Brush Name=<Name>

            Begin PolyList

                  ...

            End PolyList

      End Brush

End Actor
```

<Name>
: There is only one name for the brush, and it should be used in both places listed above.

Maps

The map format is quite simple, merely listing all the actors used in the map. Note that brushes are also represented in their actor form in this type of T3D file.

```
Begin Map

      Begin Actor Class=... Name=...

            ...

      End Actor

          Begin Actor Class=... Name=...

          ...

          End Actor

          ...

End Map
```

Polygon Flags

These are the flags referenced by polygons. A C++ header file delineating these flags as an enum is also available: enum-polyflags.h

In-game flags:

PF_Invisible, 0x00000001
: Polygon is invisible.

PF_Masked, 0x00000002
: Polygon should be drawn masked.

PF_Translucent, 0x00000004
: Polygon is transparent.

PF_NotSolid, 0x00000008
: Polygon is not solid doesn't block.

PF_Environment, 0x00000010
: Polygon should be drawn environment mapped.

PF_ForceViewZone, 0x00000010
: Force current iViewZone in OccludeBSP (reuse Environment flag)

PF_Semisolid, 0x00000020
: Polygon is semi-solid, collision solid CSG nonsolid.

PF_Modulated, 0x00000040
: Modulation transparency.

PF_FakeBackdrop, 0x00000080
: Polygon looks exactly like backdrop.

PF_TwoSided, 0x00000100
: Polygon is visible from both sides.

PF_AutoUPan, 0x00000200
: Automatically pans in U direction.

PF_AutoVPan, 0x00000400
: Automatically pans in V direction.

PF_NoSmooth, 0x00000800
: Don't smooth textures.

PF_BigWavy, 0x00001000
: Polygon has a big wavy pattern in it.

PF_SpecialPolygon, 0x00001000
: Game-specific Polygon-level render control (reuse BigWavy flag)

PF_SmallWavy, 0x00002000
: Small wavy pattern (for water/enviro reflection).

PF_Flat, 0x00004000
: Flat surface.

PF_LowShadowDetail, 0x00008000
: Low detail shadows.

PF_NoMerge, 0x00010000
: Don't merge Polygon's nodes before lighting when rendering.

PF_CloudWavy, 0x00020000
: Polygon appears wavy like clouds.

PF_DirtyShadows, 0x00040000

Dirty shadows.
PF_BrightCorners, 0x00080000
Brighten convex corners.
PF_SpecialLit, 0x00100000
Only speciallit lights apply to this Polygon.
PF_Gouraud, 0x00200000
Gouraud shaded.
PF_NoBoundRejection, 0x00200000
Disable bounds rejection in OccludeBSP (reuse Gouraud flag)
PF_Unlit, 0x00400000
Unlit.
PF_HighShadowDetail, 0x00800000
High detail shadows.
PF_Portal, 0x04000000
Portal between iZones.
PF_Mirrored, 0x08000000
Reflective surface.

Editor flags:

PF_Memorized, 0x01000000
Editor: Polygon is remembered.
PF_Selected, 0x02000000
Editor: Polygon is selected.
PF_Highlighted, 0x10000000
Editor: Polygon is highlighted.
PF_FlatShaded, 0x40000000
FPolygon has been split by SplitPolygonWithPlane.

Internal flags:

PF_EdProcessed, 0x40000000
FPolygon was already processed in editorBuildFPolygons.
PF_EdCut, 0x80000000
FPolygon has been split by SplitPolygonWithPlane.
PF_RenderFog, 0x40000000
Render with fogmapping.
PF_Occlude, 0x80000000
Occludes even if PF_NoOcclude.
PF_RenderHint, 0x01000000
Rendering optimization hint.

Combinations of flags:

PF_NoOcclude, PF_Masked | PF_Translucent | PF_Invisible | PF_Modulated

PF_NoEdit, PF_Memorized | PF_Selected | PF_EdProcessed | PF_NoMerge | PF_EdCut
PF_NoImport, PF_NoEdit | PF_NoMerge | PF_Memorized | PF_Selected | PF_EdProcessed | PF_EdCut
PF_AddLast, PF_Semisolid | PF_NotSolid
PF_NoAddToBSP, PF_EdCut | PF_EdProcessed | PF_Selected | PF_Memorized
PF_NoShadows, PF_Unlit | PF_Invisible | PF_Environment | PF_FakeBackdrop
PF_Transient, PF_Highlighted

Last updated: Wednesday, October 25, 2000 7:04:34 AM