**Q3 .map structure**

**Q3 .map structure**

The Quake .map Format is a plain text file which contains definitions of brushes and entities to be used by QBSP and it's related compiling tools to create a .bsp file used by Quake as levels. They are generally created by level editing software.

**A simple map**

{

"spawnflags" "0"

"classname" "worldspawn"

"wad" "E:\q1maps\Q.wad"

{

( 256 64 16 ) ( 256 64 0 ) ( 256 0 16 ) mmetal1\_2 0 0 0 1 1

( 0 0 0 ) ( 0 64 0 ) ( 0 0 16 ) mmetal1\_2 0 0 0 1 1

( 64 256 16 ) ( 0 256 16 ) ( 64 256 0 ) mmetal1\_2 0 0 0 1 1

( 0 0 0 ) ( 0 0 16 ) ( 64 0 0 ) mmetal1\_2 0 0 0 1 1

( 64 64 0 ) ( 64 0 0 ) ( 0 64 0 ) mmetal1\_2 0 0 0 1 1

( 0 0 -64 ) ( 64 0 -64 ) ( 0 64 -64 ) mmetal1\_2 0 0 0 1 1

}

}

{

"spawnflags" "0"

"classname" "info\_player\_start"

"origin" "32 32 24"

}

This is an example of a simple map. There is a Worldspawn entity, a special entity which contains all of a map's solid geometry as well as some map properties, which has a single brush defined, and an info\_player\_start entity.

**Structure**

{

entity

{

brush (optional)

}

}

The general structure of a .map file is to contain entity objects between { } brackets, and to nest brushes within these objects.

**Entity definition**

{

"spawnflags" "0"

"classname" "info\_player\_start"

"origin" "32 32 24"

}

An entity is simply defined by it's classname, it's origin, and it's various keys. An entity may also contain a brush, if it's of a type which uses a brush. An example of this is the worldspawn entity seen in our simple map, which contains a brush.

**Brush definition**

{

( 256 64 16 ) ( 256 64 0 ) ( 256 0 16 ) mmetal1\_2 0 0 0 1 1

( 0 0 0 ) ( 0 64 0 ) ( 0 0 16 ) mmetal1\_2 0 0 0 1 1

( 64 256 16 ) ( 0 256 16 ) ( 64 256 0 ) mmetal1\_2 0 0 0 1 1

( 0 0 0 ) ( 0 0 16 ) ( 64 0 0 ) mmetal1\_2 0 0 0 1 1

( 64 64 0 ) ( 64 0 0 ) ( 0 64 0 ) mmetal1\_2 0 0 0 1 1

( 0 0 -64 ) ( 64 0 -64 ) ( 0 64 -64 ) mmetal1\_2 0 0 0 1 1

}

A brush consists of several faces, each of which is defined by a plane and additional texture information. The actual geometry of the brush is only defined by the planes, of which there must be at least four. Each plane defines a half space, that is, an infinite set of points that is bounded by a plane. The intersection of these half spaces forms a convex polyhedron. Note that the actual vertices of the brush are **NOT** stored in the map file. The vertices are determined by BSP (or a level editing program) by computing the intersections of the planes.

In the example shown here, this brush is a 6 sided cuboid. The plane of its first face is defined by 3 points, ( 256 64 16 ) ( 256 64 0 ) ( 256 0 16 ). The other information supplied is the texture used by the face. "mmetal1\_2" is the name of the texture, a single plane may only have a single texture. "0 0 0 1 1" are how the texture is display, and are respectively "X offset" "Y offset" "Rotation" "X scale" and "Y scale".

The plane points ( p1 ) ( p2 ) ( p3 ) are interpreted as follows. The plane points must be arranged such that the cross product of the vectors (p3 - p1) and (p2 - p1) is not null, that is, the three points must be linearly independent. Then, the normalized cross product represents the normal vector of the plane. Every point p for which (p - p1) \* normal <= 0 (where \* is the dot product) holds is considered to be in the half space defined by the plane. Every other point is considered not to be in the half space.

The intersection of half spaces interpretation of brushes does not yield the vertices, however. While there are efficient methods to compute them directly, the QBSP compiler uses a different approach. Starting with the edges and vertices of a huge cube (usually ranging from -4096 to 4096 in each dimension), the edges of the cube are intersected with the planes of the faces. Upon each intersection, new edges and vertices are generated that replace some of the old ones until all planes have been used. Note that due to floating point inaccuracies, there may be slight errors in the position of each vertex. This may lead to QBSP warnings when it heals degenerate vertices and edges.

In the example used here, all plane points have integer coordinates. In the original Quake map format, only integer coordinates are allowed. This has some advantages and drawbacks. The major advantage is that integer numbers can be written to text files without loss of precision, which is not always true for floating point numbers. The major drawback is that not every plane can be accurately represented using integer coordinates for its points. This becomes problematic when brushes are rotated arbitrarily or when the vertices are edited directly. That is why most modern Quake compilers allow floating point coordinates for plane points.