Unofficial Quake 3 Map Specs

Introduction [top]

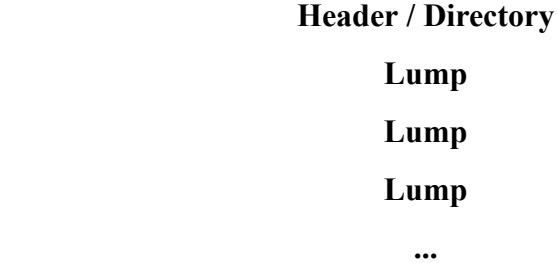
This document describes the Quake 3 BSP file format. This is an unofficial document. Quake 3 is a registered trademark of id Software, which does not sponsor, authorize, or endorse this document.

This document describes the Quake 3 BSP file format as the author understands it. While every effort has been made to ensure that the contents of this document are accurate, the author does not guarantee that any portion of this document is actually correct. In addition, the author cannot be held responsible the consequences of the any use or misuse of the information contained in this document.

Description [top]

File structure

Quake 3 BSP files are IBSP files, and therefore have a structure similar to previous BSP files from id Software. Every IBSP file begins with a header, which in turn contains a lump directory. The lump directory describes the layout of the rest of the file, which contains some number of lumps. Each lump stores a particular kind of map data.



The layout of an IBSP file. *An IBSP file consists of a header followed by a number of lumps. The header contains a directory which identifies the locations and sizes of the lumps.*

Data types

Quake 3 BSP files contains only four basic data types. They are:

Type	Description
ubyte	unsigned byte
int	4-byte integer, little-endian
float	4-byte IEEE float, little-endian
string[*n*]	string of *n* ASCII bytes, not necessarily null-terminated

All data in a BSP file is organized into records composed of these four data types.

Header and Directory

The header record looks like this:

header

string[4] *magic*	Magic number. Always "IBSP".
int *version*	Version number. 0x2e for the BSP files distributed with Quake 3.
direntry[17] *direntries*	Lump directory, seventeen entries.

Each **direntry** locates a single lump in the BSP file:

direntry

int *offset*	Offset to start of lump, relative to beginning of file.
int *length*	Length of lump. Always a multiple of 4.

Lumps

There are 17 lumps in a Quake 3 BSP file. In the order that they appear in the lump directory, they are:

Index	Lump Name	Description
0	Entities	Game-related object descriptions.
1	Textures	Surface descriptions.
2	Planes	Planes used by map geometry.
3	Nodes	BSP tree nodes.
4	Leafs	BSP tree leaves.
5	Leaffaces	Lists of face indices, one list per leaf.
6	Leafbrushes	Lists of brush indices, one list per leaf.
7	Models	Descriptions of rigid world geometry in map.
8	Brushes	Convex polyhedra used to describe solid space.
9	Brushsides	Brush surfaces.
10	Vertexes	Vertices used to describe faces.
11	Meshverts	Lists of offsets, one list per mesh.
12	Effects	List of special map effects.
13	Faces	Surface geometry.
14	Lightmaps	Packed lightmap data.
15	Lightvols	Local illumination data.
16	Visdata	Cluster-cluster visibility data.

Entities

The entities lump stores game-related map information, including information about the map name, weapons, health, armor, triggers, spawn points, lights, and .md3 models to be placed in the map. The lump contains only one record, a string that describes all of the entities:

entities

string[*length*] *ents*	Entity descriptions, stored as a string.

The *length* of the entity string is given by the size of the lump itself, as specified in the lump directory.

The meanings, formats, and parameters of the various entity descriptions are currently outside the scope of this document. For more information about entity descriptions, see the documentation to Q3Radiant, the Quake 3 level editor.

Textures

The textures lump stores information about surfaces and volumes, which are in turn associated with faces, brushes, and brushsides. There are a total of *length* / sizeof(**texture**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

texture

string[64] *name*	Texture name.

int *flags*	Surface flags.
int *contents*	Content flags.

Planes

The planes lump stores a generic set of planes that are in turn referenced by nodes and brushsides. There are a total of *length* / sizeof(**plane**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

plane

float[3] *normal*	Plane normal.
float *dist*	Distance from origin to plane along normal.

Note that planes are paired. The pair of planes with indices i and i ^ 1 are coincident planes with opposing normals.

Nodes

The nodes lump stores all of the nodes in the map's BSP tree. The BSP tree is used primarily as a spatial subdivision scheme, dividing the world into convex regions called leafs. The first node in the lump is the tree's root node. There are a total of *length* / sizeof(**node**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

node

int *plane*	Plane index.
int[2] *children*	Children indices. Negative numbers are leaf indices: -(leaf+1).
int[3] *mins*	Integer bounding box min coord.
int[3] *maxs*	Integer bounding box max coord.

Leafs

The leafs lump stores the leaves of the map's BSP tree. Each leaf is a convex region that contains, among other things, a cluster index (for determining the other leafs potentially visible from within the leaf), a list of faces (for rendering), and a list of brushes (for collision detection). There are a total of *length* / sizeof(**leaf**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

leaf

int *cluster*	Visdata cluster index.
int *area*	Areaportal area.
int[3] *mins*	Integer bounding box min coord.
int[3] *maxs*	Integer bounding box max coord.
int *leafface*	First leafface for leaf.
int *n_leaffaces*	Number of leaffaces for leaf.
int *leafbrush*	First leafbrush for leaf.
int *n_leafbrushes*	Number of leafbrushes for leaf.

If *cluster* is negative, the leaf is outside the map or otherwise invalid.

Leaffaces

The leaffaces lump stores lists of face indices, with one list per leaf. There are a total of *length* / sizeof(**leafface**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

leafface

int *face*	Face index.

Leafbrushes

The leafbrushes lump stores lists of brush indices, with one list per leaf. There are a total of *length* / sizeof(**leafbrush**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

leafbrush

int *brush*	Brush index.

Models

The models lump describes rigid groups of world geometry. The first model correponds to the base portion of the map while the remaining models correspond to movable portions of the map, such as the map's doors, platforms, and buttons. Each model has a list of faces and list of brushes; these are especially important for the movable parts of the map, which (unlike the base portion of the map) do not have BSP trees associated with them. There are a total of *length* / sizeof(**models**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

model

float[3] *mins*	Bounding box min coord.
float[3] *maxs*	Bounding box max coord.
int *face*	First face for model.
int *n_faces*	Number of faces for model.
int *brush*	First brush for model.
int *n_brushes*	Number of brushes for model.

Brushes

The brushes lump stores a set of brushes, which are in turn used for collision detection. Each brush describes a convex volume as defined by its surrounding surfaces. There are a total of *length* / sizeof(**brushes**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

brush

int *brushside*	First brushside for brush.
int *n_brushsides*	Number of brushsides for brush.
int *texture*	Texture index.

Brushsides

The brushsides lump stores descriptions of brush bounding surfaces. There are a total of *length* / sizeof(**brushsides**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

brushside

int *plane*	Plane index.
int *texture*	Texture index.

Vertexes

The vertexes lump stores lists of vertices used to describe faces. There are a total of *length* / sizeof(**vertex**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

vertex

float[3] *position*	Vertex position.
float[2][2] *texcoord*	Vertex texture coordinates. 0=surface, 1=lightmap.
float[3] *normal*	Vertex normal.
ubyte[4] *color*	Vertex color. RGBA.

Meshverts

The meshverts lump stores lists of vertex offsets, used to describe generalized triangle meshes. There are a total of *length* / sizeof(**meshvert**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

meshvert

int *offset*	Vertex index offset, relative to first vertex of corresponding face.

Effects

The effects lump stores references to volumetric shaders (typically fog) which affect the rendering of a particular group of faces. There are a total of *length* / sizeof(**effect**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

effect

string[64] *name*	Effect shader.
int *brush*	Brush that generated this effect.
int *unknown*	Always 5, except in q3dm8, which has one effect with -1.

Faces

The faces lump stores information used to render the surfaces of the map. There are a total of *length* / sizeof(**faces**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

face

int *texture*	Texture index.
int *effect*	Index into lump 12 (Effects), or -1.
int *type*	Face type. 1=polygon, 2=patch, 3=mesh, 4=billboard
int *vertex*	Index of first vertex.
int *n_vertexes*	Number of vertices.
int *meshvert*	Index of first meshvert.
int *n_meshverts*	Number of meshverts.
int *lm_index*	Lightmap index.
int[2] *lm_start*	Corner of this face's lightmap image in lightmap.
int[2] *lm_size*	Size of this face's lightmap image in lightmap.
float[3] *lm_origin*	World space origin of lightmap.
float[2][3] *lm_vecs*	World space lightmap s and t unit vectors.
float[3] *normal*	Surface normal.
int[2] *size*	Patch dimensions.

There are four types of faces: polygons, patches, meshes, and billboards.

Several components have different meanings depending on the face type.

For type 1 faces (polygons), *vertex* and *n_vertexes* describe a set of vertices that form a polygon. The set always contains a loop of vertices, and sometimes also includes an additional vertex near the center of the polygon. For these faces, *meshvert* and *n_meshverts* describe a valid polygon triangulation. Every three meshverts describe a triangle. Each meshvert is an offset from the first vertex of the face, given by *vertex*.

For type 2 faces (patches), *vertex* and *n_vertexes* describe a 2D rectangular grid of control vertices with dimensions given by *size*. Within this rectangular grid, regions of 3×3 vertices represent biquadratic Bezier patches. Adjacent patches share a line of three vertices. There are a total of (*size*[0] - 1) / 2 by (*size*[1] - 1) / 2 patches. Patches in the grid start at (i, j) given by:

i = 2n, n in [0 .. (*size*[0] - 1) / 2), and
j = 2m, m in [0 .. (*size*[1] - 1) / 2).

For type 3 faces (meshes), *meshvert* and *n_meshverts* are used to describe the independent triangles that form the mesh. As with type 1 faces, every three meshverts describe a triangle, and each meshvert is an offset from the first vertex of the face, given by *vertex*.

For type 4 faces (billboards), *vertex* describes the single vertex that determines the location of the billboard. Billboards are used for effects such as flares. Exactly how each billboard vertex is to be interpreted has not been investigated.

The *lm_* variables are primarily used to deal with lightmap data. A face that has a lightmap has a non-negative *lm_index*. For such a face, *lm_index* is the index of the image in the lightmaps lump that contains the lighting data for the face. The data in the lightmap image can be located using the rectangle specified by *lm_start* and *lm_size*.

For type 1 faces (polygons) only, *lm_origin* and *lm_vecs* can be used to compute the world-space positions corresponding to lightmap samples. These positions can in turn be used to compute dynamic lighting across the face.

None of the *lm_* variables are used to compute texture coordinates for indexing into lightmaps. In fact, lightmap coordinates need not be computed. Instead, lightmap coordinates are simply stored with the vertices used to describe each face.

Lightmaps

The lightmaps lump stores the light map textures used make surface lighting look more realistic. There are a total of *length* / sizeof(**lightmap**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

lightmap

ubyte[128][128][3] *map*	Lightmap color data. RGB.

Lightvols

The lightvols lump stores a uniform grid of lighting information used to illuminate non-map objects. There are a total of *length* / sizeof(**lightvol**) records in the lump, where *length* is the size of the lump itself, as specified in the lump directory.

Lightvols make up a 3D grid whose dimensions are:

nx = floor(models[0].maxs[0] / 64) - ceil(models[0].mins[0] / 64) + 1
ny = floor(models[0].maxs[1] / 64) - ceil(models[0].mins[1] / 64) + 1
nz = floor(models[0].maxs[2] / 128) - ceil(models[0].mins[2] / 128) + 1

lightvol

ubyte[3] *ambient*	Ambient color component. RGB.
ubyte[3] *directional*	Directional color component. RGB.
ubyte[2] *dir*	Direction to light. 0=phi, 1=theta.

Visdata

The visdata lump stores bit vectors that provide cluster-to-cluster visibility information. There is exactly one visdata record, with a *length* equal to that specified in the lump directory.

visdata

int *n_vecs*	Number of vectors.
int *sz_vecs*	Size of each vector, in bytes.
ubyte[*n_vecs* * *sz_vecs*] *vecs*	Visibility data. One bit per cluster per vector.

Cluster x is visible from cluster y if the (1 << y % 8) bit of *vecs*[x * *sz_vecs* + y / 8] is set.

Note that clusters are associated with leaves.

Known Issues and Missing Items [top]

This document is very brief. I have gathered more information, but have not had time to write it up. Occasionally, I add more information to this document.

At some point I put together a page that describes triangle meshes and other q3 leaf elements. I forget the exact reason I created that page, but you might find it interesting.

Feel free to ask for clarification, but please accept my apologies if I can't find the time to answer.

Keywords: quake 3 quake3 q3 arena quake3arena q3arena map bsp file spec specs format