VALVE Developer Community

Create account Log in

Navigation
Main Page
Source SDK index
Recent changes
Random page

Support
Getting help
Source SDK FAQ
Level Design FAQ
SDK Help Forums

Steam Community
Source SDK Hub
Steam Games

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information

Page Discussion View View source History Search

Valve Texture Format

The **Valve Texture Format** (**VTF**) is the proprietary texture format used by the Source engine. VTF files are generally referenced in a Material instead of being accessed directly, which allows re-use in different ways.

VTF files can be created from TGA images using the Source SDK Tool VTEX, or from most common image formats with third-party tools. Both textures and materials are stored in subfolders of `game_dir/materials/`.

VTF files must have dimensions that are powers of two.

Contents [hide]

Storage capabilities

The VTF image format can store either a flat texture, an environment map, or a volumetric texture. Each of these can have multiple frames.

- An environment map is a six-faced cube map
- A volumetric texture is a texture with depth, where each frame is a "layer" which are layered in the third dimension. So a 16x16x16 volumetric texture has 16 separate 16x16 textures stacked to give depth. This format is used

internally by Source, and you shouldn't have any need to actually create one yourself.

- For each frame and face, the VTF file contains both the basic original source-image data (pixel map) and a series of mipmaps used for rendering the texture over varying distances. Because each successive mipmap is exactly 1/2 the dimension (height and width) of the previous one, the source-image dimensions must be powers of 2. Although the source-image may be rectangular, square mipmaps are stored more efficiently in the VTF.
- Start frame (for animations)
- Bump map scale
- A Reflectivity value for use by VRAD
- A very low resolution copy of the VTF for color sampling by the engine.

Resources

VTF 7.3 added an extensible "resource data" system. Anything can be added, but Source will recognise only the following:

- A CRC value, for detecting data corruption.
- An U/V LOD control. This is the highest mipmap which should be loaded when game's Texture Detail setting is "High" (`mat_picmip 0`). An U LOD Control value of 11 selects the mipmap which is 2048 pixels (2^{11}) across.

  Since users are currently only presented with one texture detail setting above High, there is little point setting this value to anything except 50% or 100% of your texture's size.
- Animated particle sheet data.
- Expanded texture settings. This is a collection of 32 flags, none of which are in use by Valve. Unlike the built-in VTF flags these can be defined on a game-by-game basis.

Image data formats

The VTF image format can store image data in a variety of formats. Some formats were meant for the engine, some only as an interim format for conversions. The uncompressed formats are not lossy and the compressed (DXT) formats are.

Image data format table

Format	Red Bits	Green Bits	Blue Bits	Alpha Bits	Total Bits	Compressed	Supported	Comments
A8	0	0	0	8	8	False	True	
ABGR8888	8	8	8	8	32	False	True	Uncompressed texture with alpha
ARGB8888	8	8	8	8	32	False	True	
BGR565	5	6	5	0	16	False	True	Uncompressed texture, limited color depth
BGR888	8	8	8	0	24	False	True	Uncompressed texture
BGR888_BLUESCREEN	8	8	8	0	24	False	True	
BGRA4444	4	4	4	4	16	False	True	Uncompressed texture with alpha, half color depth
BGRA5551	5	5	5	1	16	False	True	
BGRA8888	8	8	8	8	32	*Either*	True	Also used for compressed HDR
BGRX5551	5	5	5	1	16	False	True	
BGRX8888	8	8	8	8	32	False	True	
DXT1	N/A	N/A	N/A	0	4	**True**	True	Standard compression, no alpha
								Standard

DXT1_ONEBITALPHA	N/A	N/A	N/A	1	4	**True**	True	compression, one bit alpha
DXT3	N/A	N/A	N/A	4	8	**True**	True	Uninterpolated Alpha
DXT5	N/A	N/A	N/A	4	8	**True**	True	Interpolated Alpha (recommended)
I8	N/A	N/A	N/A	N/A	8	False	True	Luminance (Grayscale)
IA88	N/A	N/A	N/A	8	16	False	True	Luminance (Grayscale)
P8	N/A	N/A	N/A	N/A	8	False	**False**	Paletted
RGB565	5	6	5	0	16	False	True	
RGB888	8	8	8	0	24	False	True	
RGB888_BLUESCREEN	8	8	8	0	24	False	True	
RGBA16161616	16	16	16	16	64	False	True	Integer HDR Format
RGBA16161616F	16	16	16	16	64	False	True	Floating Point HDR Format
RGBA8888	8	8	8	8	32	False	True	
UV88	N/A	N/A	N/A	N/A	16	False	True	Uncompressed du/dv Format
UVLX8888	N/A	N/A	N/A	N/A	32	False	True	
UVWQ8888	N/A	N/A	N/A	N/A	32	False	True	

HDR compression

HDR compression

HDR textures can be stored in compressed form using the BGRA8888 format.

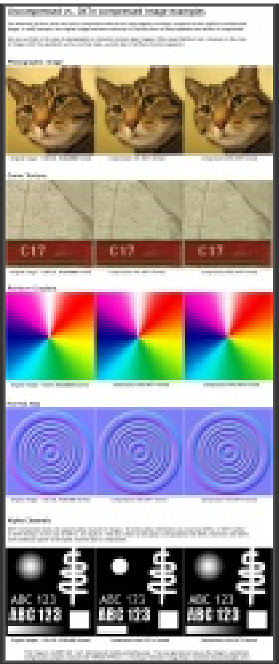
The formula to convert these colors back to integer HDR is:

```
RGB = RGB * (A * 16)
```

and for floating point HDR:

```
RGB = (RGB * (A * 16)) / 262144
```

Choosing an image format


Comparison of how DXTn compression affects different types of images

Though the VTF image format provides support for a wide range of image data formats, there are only a handful of image data formats you are likely to use. These formats and their criteria are described below:

- **BGR888:** use this format for textures with no alpha channel and very fine gradients (i.e. normal maps or light halos).
- **BGRA8888:** use this format for textures with an alpha channel and very fine gradients (i.e. normal maps or light halos). It can also be used to produce Very High quality textures.
- **DXT1:** use this format for typical textures with no alpha channel.
- **DXT3:** use this format for typical textures with an alpha channel with sharp gradients.
- **DXT5:** use this format for typical textures with an alpha channel with smooth gradients.
- **I8:** use this format for black and white textures with no alpha channel and very fine gradients (i.e. light halos).
- **IA88:** use this format for black and white textures with an alpha channel and very fine gradients (i.e. smoke or light halos).
- **RGBA16161616F:** use this format for HDR textures.
- **UV88:** use this format for DuDv maps.

Find technical details on the various DXT compression formats here.

Image flags

Tip: Most shader settings are configured as material parameters, not texture flags.

A VTF file can contain the following flags (version 7.5):

Flag	Value	Comment
Point Sampling	0x0001	Low quality, "pixel art" texture filtering.
Trilinear Sampling	0x0002	Medium quality texture filtering.
Clamp S	0x0004	Clamp S coordinates.
Clamp T	0x0008	Clamp T coordinates.
Anisotropic Sampling	0x0010	High quality texture filtering.
Hint DXT5	0x0020	Used in skyboxes. Makes sure edges are seamless.
PWL Corrected	0x0040	Purpose unknown.
SRGB	n/a	Uses space RGB. Useful for High Gamuts. Deprecated in 7.5.
No Compress	0x0040	No DXT compression used. Deprecated
Normal Map	0x0080	Texture is a normal map.
No Mipmaps	0x0100	Render largest mipmap only. (Does not delete existing mipmaps, just disables them.)
No Level Of Detail	0x0200	Not affected by texture resolution settings.
No Minimum Mipmap	0x0400	
Procedural	0x0800	Texture is an procedural texture (code can modify it).
One Bit Alpha	0x1000	One bit alpha channel used.
Eight Bit Alpha	0x2000	Eight bit alpha channel used.
Environment Map	0x4000	Texture is an environment map.

Render Target	0x8000	Texture is a render target.
Depth Render Target	0x10000	Texture is a depth render target.
No Debug Override	0x20000	
Single Copy	0x40000	
Pre SRGB	0x80000	SRGB correction has already been applied
One Over Mipmap Level In Alpha	0x80000	Fill the alpha channel with 1/Mipmap Level. Deprecated (Internal to VTEX?)
Premultiply Color By One Over Mipmap Level	0x100000	(Internal to VTEX?)
Normal To DuDv	0x200000	Texture is a DuDv map. (Internal to VTEX?)
Alpha Test Mipmap Generation	0x400000	(Internal to VTEX?)
No Depth Buffer	0x800000	Do not buffer for Video Processing, generally render distance.
Nice Filtered	0x1000000	Use NICE filtering to generate mipmaps. (Internal to VTEX?)
Clamp U	0x2000000	Clamp U coordinates (for volumetric textures).
Vertex Texture	0x4000000	Usable as a vertex texture
SSBump	0x8000000	Texture is a SSBump. (*SSB*)
Border	0x20000000	Clamp to border colour on all texture coordinates

File format

The VTF image format is described as follows.

VTF layout

7.2	7.3 +
1. VTF Header	1. VTF Header
2. VTF Low Resolution Image Data	2. Resource entries
3. For Each Mipmap (Smallest to Largest)	■ VTF Low Resolution Image Data
■ For Each Frame (First to Last)	■ Other resource data
■ For Each Face (First to Last)	■ For Each Mipmap (Smallest to Largest)
■ For Each Z Slice (Min to Max; Varies with Mipmap)	■ For Each Frame (First to Last)
■ VTF High Resolution Image Data	■ For Each Face (First to Last)
	■ For Each Z Slice (Min to Max; Varies with Mipmap)
	■ VTF High Resolution Image Data

VTF enumerations

```
enum
{
        IMAGE_FORMAT_NONE = -1,
        IMAGE_FORMAT_RGBA8888 = 0,
        IMAGE_FORMAT_ABGR8888,
        IMAGE_FORMAT_RGB888,
        IMAGE_FORMAT_BGR888,
        IMAGE_FORMAT_RGB565,
        IMAGE_FORMAT_I8,
        IMAGE_FORMAT_IA88,
        IMAGE_FORMAT_P8,
        IMAGE_FORMAT_A8,
        IMAGE_FORMAT_RGB888_BLUESCREEN,
        IMAGE_FORMAT_BGR888_BLUESCREEN,
        IMAGE_FORMAT_ARGB8888,
        IMAGE_FORMAT_BGRA8888,
        IMAGE_FORMAT_DXT1,
        IMAGE_FORMAT_DXT3,
        IMAGE_FORMAT_DXT5,
        IMAGE_FORMAT_BGRX8888,
        IMAGE_FORMAT_BGR565,
        IMAGE_FORMAT_BGRX5551,
        IMAGE_FORMAT_BGRA4444,
        IMAGE_FORMAT_DXT1_ONEBITALPHA,
        IMAGE_FORMAT_BGRA5551,
        IMAGE_FORMAT_UV88,
        IMAGE_FORMAT_UVWQ8888,
        IMAGE_FORMAT_RGBA16161616F,
        IMAGE_FORMAT_RGBA16161616,
        IMAGE_FORMAT_UVLX8888
};
```

```
enum
{
        // Flags from the *.txt config file
        TEXTUREFLAGS_POINTSAMPLE = 0x00000001,
        TEXTUREFLAGS_TRILINEAR = 0x00000002,
        TEXTUREFLAGS_CLAMPS = 0x00000004,
        TEXTUREFLAGS_CLAMPT = 0x00000008,
        TEXTUREFLAGS_ANISOTROPIC = 0x00000010,
        TEXTUREFLAGS_HINT_DXT5 = 0x00000020,
        TEXTUREFLAGS_PWL_CORRECTED = 0x00000040,
        TEXTUREFLAGS_NORMAL = 0x00000080,
        TEXTUREFLAGS_NOMIP = 0x00000100,
        TEXTUREFLAGS_NOLOD = 0x00000200,
        TEXTUREFLAGS_ALL_MIPS = 0x00000400,
        TEXTUREFLAGS_PROCEDURAL = 0x00000800,

        // These are automatically generated by vtex from the texture data
        TEXTUREFLAGS_ONEBITALPHA = 0x00001000,
        TEXTUREFLAGS_EIGHTBITALPHA = 0x00002000,

        // Newer flags from the *.txt config file
        TEXTUREFLAGS_ENVMAP = 0x00004000,
        TEXTUREFLAGS_RENDERTARGET = 0x00008000,
        TEXTUREFLAGS_DEPTHRENDERTARGET = 0x00010000,
        TEXTUREFLAGS_NODEBUGOVERRIDE = 0x00020000,
        TEXTUREFLAGS_SINGLECOPY = 0x00040000,
        TEXTUREFLAGS_PRE_SRGB = 0x00080000,

        TEXTUREFLAGS_UNUSED_00100000 = 0x00100000,
        TEXTUREFLAGS_UNUSED_00200000 = 0x00200000,
        TEXTUREFLAGS_UNUSED_00400000 = 0x00400000,

        TEXTUREFLAGS_NODEPTHBUFFER = 0x00800000,

        TEXTUREFLAGS_UNUSED_01000000 = 0x01000000,

        TEXTUREFLAGS_CLAMPU = 0x02000000,
        TEXTUREFLAGS_VERTEXTEXTURE = 0x04000000,
        TEXTUREFLAGS_SSBUMP = 0x08000000,

        TEXTUREFLAGS_UNUSED_10000000 = 0x10000000,

        TEXTUREFLAGS_BORDER = 0x20000000,

        TEXTUREFLAGS_UNUSED_40000000 = 0x40000000,
        TEXTUREFLAGS_UNUSED_80000000 = 0x80000000,
};
```

VTF header

```
typedef struct tagVTFHEADER
{
        char            signature[4];           // File signature ("VTF\0"
        unsigned int    version[2];             // version[0].version[1] (
        unsigned int    headerSize;             // Size of the header stru
        unsigned short  width;                  // Width of the largest mi
        unsigned short  height;                 // Height of the largest m
        unsigned int    flags;                  // VTF flags.
        unsigned short  frames;                 // Number of frames, if an
        unsigned short  firstFrame;             // First frame in animatio
        unsigned char   padding0[4];            // reflectivity padding (1
        float           reflectivity[3];        // reflectivity vector.
        unsigned char   padding1[4];            // reflectivity padding (8
        float           bumpmapScale;           // Bumpmap scale.
        unsigned int    highResImageFormat;     // High resolution image f
        unsigned char   mipmapCount;            // Number of mipmaps.
        unsigned int    lowResImageFormat;      // Low resolution image fo
        unsigned char   lowResImageWidth;       // Low resolution image wi
        unsigned char   lowResImageHeight;      // Low resolution image he

        // 7.2+
        unsigned short  depth;                  // Depth of the largest mi
                                                // Must be a power of 2. C

        // 7.3+
        unsigned char   padding2[3];            // depth padding (4 byte a
        unsigned int    numResources;           // Number of resources thi
} VTFHEADER;
```

VTF lo-res image data

Tightly packed low resolution image data in the format described in the header. The low resolution image data is always stored in the DXT1 compressed image format. Its dimensions are that of the largest mipmap with a width or height that does not exceed 16 pixels. i.e. for a 256x256 pixel VTF: 16x16, for a 256x64 pixel VTF: 16x4, for a 1x32 pixel VTF: 1x16, for a 4x4 pixel VTF: 4x4.

VTF hi-res image data

Tightly packed interleaved high resolution image data in the format described in the header. Common image formats include DXT1, DXT5, BGR888, BGRA8888 and UV88. All dimensions must be a power of two.

Version history

v7.5

- Released July 19th, 2010 as part of Alien Swarm
- Bitwise equivalent to v7.4.
- Redefines and revises two texture flags.
- Spheremaps now officially redundant.
- Most changes internal to the VTF creation process with VTEX, e.g. MipMap fading, Alpha decay and XBox360 formats.

v7.4

- Released October 10th, 2007 as part of The Orange Box.
- Bitwise equivalent to v7.3.
- Addresses issues related to how gamma-correction is performed on textures for TV-output on XBOX 360 combined with hunting down OS Paged Pool Memory.

v7.3

- Added an extensible resource orientated structure.
- Added CRC, Texture LOD Control and Sheet resources, along with backwards compatible Image and Low Resolution Image resources.
- Added several vendor specific depth-stencil formats (for internal engine use), along with normal map formats and linear uncompressed formats.
- Released September 18th, 2007 as part of the Team Fortress 2 beta.

v7.2

- Added volumetric texture support.
- Released September 23rd, 2005 as a Steam engine update.

v7.1

- Added spheremap support to environment maps. (This was intended for DirectX 6 support which was later cut.)

v7.0

- Initial release. (Internal release only, however, some v7.0 textures made it to the published title.)

Implementation

An example Steam independent implementation of the VTF image file format can be found in the LGPL C/C++ library VTFLib.

Utilities

Viewing

Windows thumbnail handler (Windows XP+)
VTF Explorer (Windows; can explore inside GCFs)
gdk-pixbuf-vtf (Gnome\Nautilus\Eog)
IrfanView plugin
Leadworks Image Viewer(supports VTF) [dead link]

Editing

VTFEdit (GUI) and VTFCmd (command line)
Photoshop Plug-in (6 and up)
GIMP Plug-in
Paint.NET Plug-in
3DSMax plugin (versions 6 to 2009)

Converting Versions

7.5 to 7.4 Batch Converter

See also

- Creating a Material
- Valve Material Type

Categories: Glossary | Material System | File formats