This guide was put together by Yogensia and Rafaël De Jongh and aims to explain VMT ("Valve Material Type") files in the Source engine, how to format them, and which are the most common and useful parameters to make you materials look their best in Source.

Important: The information in this guide has been tested against **Left 4 Dead 2** and **Counter-Strike: Global Offensive**. Other versions of the engine may have different support for some of the commands.

Contents:

Formatting:

Most VMT generators will generate code that works just fine but it's not as readable as it should. They tend to put absolutely everything between quotes, which is unnecessary, and breaks syntax highlighting in text editors.

Lines should be indented with tabs, and the values should be aligned with spaces or not aligned at all and separated by one space. Aligning the values with tabs is bad practice because depending on the text editor and the tab size it uses, the alignment may or may not work.

Here's some more detail regarding formatting for parameters and their values:

Parameters/Variables

Parameters themselves and variables should not be quoted.

Example: `$envmap env_cubemap`

Boolean parameters

Boolean parameters accept values 1 or 0, meaning on and off respectively. Their value should not be quoted.

Example: `$phong 1`

String/Path parameters

String parameters accept text as a value, in the case of VMT shaders usually a path. Paths in VMT files are relative to "gamedir/materials/".

Strings and Paths should always be quoted for consistency, although lack of quotes won't break them as long as there are no spaces.

File extensions in paths should be omitted.

Example: `$basetexture "models/props/garbage"`

Float parameters

Float parameters accept any numerical value, such as 10, 0.5... Numerical values should not be quoted.

Note: The leading zero on decimal values can be omitted if quotes are used but this is not necessary, so for consistency it should be avoided.

Example: `$phongboost 0.48`

Matrix parameters

Matrix parameters accept numerical values, but only in "[X Y Z]" format, X, Y and Z being the numbers. The quotes and brackets can not be omitted. Leading zeroes in this case can be safely omitted since quotes are needed regardless.

Note: The numbers inside a matrix are never quoted, quotes should only be around the brackets.

Example: `$phongfresnelranges "[1 4 6]"`

Shaders

VMT files allow us to use several shaders depending on the kind of material we want to achieve. In this guide we will focus on parameters meant to be used with the `Vertexlitgeneric` shader which is the most common shader used on models like weapons and props.

Main Parameters

`$basetexture "path/to/diffuse"` (string)

Defines an Albedo texture (also known as Diffuse), which controls the color.

`$bumpmap "path/to/normal"` (string)

Defines a Normal map texture, which controls the direction each pixel will be facing in 3D tangent space, allowing to create the illusion of more surface detail.

Phong Parameters

`$phong 1` (boolean)

Enables Phong shading.

`$phongboost 0.84` (float)

Increases Phong intensity. A value of 0 is the same as no value.

`$phongfresnelranges "[1 4 6]"` (XYZ matrix)

Defines the brightness of the phong highlight depending on the angle of incidence relative to the player viewpoint.

This option uses a matrix value, "[X Y Z]" where X controls the brightness of surfaces facing directly to the player, Z for the surfaces facing 90 degrees away (grazing angle) and Y for in between.

`$phongtint "[.8 .8 1]"` (RGB matrix)

Controls the color of the Phong highlight. Similar to the previous command, this one uses a matrix value, but this time for Red, Green and Blue values: "[R G B]".

A simple way of tweaking this is starting with the values "[1 1 1]" and reducing from there. For example, "[1 1 .8]" would tint the highlight slightly yellow.

Note: The values in `$phongtint` default to "[0 0 0]", which means that using "[1 1 1]" will increase overall Phong intensity. This shouldn't be an issue though, since Phong highlights are usually very dim by default.

`$phongexponent 24` (float)

Controls the tightness of the Phong highlight. This parameter should only be used if there's no exponent map available. To keep the Phong highlights from introducing aliasing the value should be not exceed 150.

Warning: This parameter will override `$phongexponenttexture` completely!

`$phongexponenttexture "path/to/exponent"` (string)

Specifies a custom texture that controls the tightness of the highlight. Only the Red channel in that texture will be used for this purpose. Pure white equals a `$phongexponent` value of 150 (smaller highlights), while Black equals a value of 1.

Note: `$phongexponenttexture` will only use the texture's Red channel, but note that the Green channel *will* be used to mask `$phongalbedotint`, and the Alpha channel *can* be used to mask `$rimlight`.

Warning: This parameter will be ignored if `$phongexponent` is used!

`$phongalbedotint 1` (boolean)

Allows the color of the Albedo texture to tint the Phong highlights. This is a relatively new parameter that only works in newer versions of the engine such as in CS:GO.

If present, the Green channel from `$phongexponenttexture` will be used to determine the intensity of the effect per-pixel.

Doesn't work in L4D2.

`$phongalbedoboost 48` (float)

Increases PhongAlbedoTint brightness. A value of 0 is the same as no value.

`$normalmapalphaphongmask 1` (boolean)

Forces the Phong mask to be read from the `$bumpmap` alpha channel.

`$basemapalphaphongmask 0` (boolean)

Forces the Phong mask to be read from the `$basetexture` alpha channel.

`$phongwarptexture "path/to/phongwarp"` (string)

Uses a texture to tint the phong highlight, allowing to create iridescence and other fancy effects.

This parameter may not work when `$phongexponenttexture` is being used.

Doesn't work in L4D2.

`$lightwarptexture "path/to/lightwarp"` (string)

Loads a gradient texture that tints texels depending on their brightness. This allows material color correction and can be used to achieve other effects that depend on subtle shading variations, such as toon shading.

Environment Map Parameters

`$envmap env_cubemap` (string/variable)

Specifies a cubemap used to render Environment reflections on the material.

Using the value `env_cubemap` will let the level choose the cubemap closest to the player. This is usualy the recommended setting. Alternatively, a custom cubemap texture can be specified.

Environment reflections in the Source engine heavily rely on Phong settings. The more intense the the Phong highlight is, the more intense the Environment reflections will be. A useful command to control Environment reflections independently of Phong settings is to use `$envmaptint` (see below).

`$envmaptint "[.075 .075 .125]"` (RGB matrix)

Tints the Environment reflections color.

Tip: Note that unlike `$phongtint`, this parameter defaults to "[1 1 1]", which means that using a very low value will dim the Environment reflections without affecting the Phong settings. For example, a value of "[.02 .02 .02]" will greatly reduce the Environment reflections. This may be useful in many cases, specially when trying to represent materials with low glossiness.

`$envmapfresnel 1` (float)

Adds a Fresnel term to Environment reflections, which increases their intensity depending on the angle of incidence. Areas facing away from the player's point of view will be more reflective.

Increasing this value increases the Fresnel effect but it will also increase the overall Environment reflections intensity. Again, this can be tweaked in combination with `$envmaptint` to avoid excessive over-brightening.

`$basealphaenvmapmask 0` (boolean)

Forces the Environment reflections mask to be read from the `$basetexture` alpha channel.

`$normalmapalphaenvmapmask 1` (boolean)

Forces the Environment reflections mask to be read from the `$bumpmap` alpha channel.

Rim Light

`$rimlight 1` (boolean)

Enables Rim Lighting, adding a subtle of ambient light to the model.

Doesn't work in L4D2.

`$rimlightexponent 24` (float)

Controls the width of the Rim highlight.

`$rimlightboost 1.2` (float)

Increases Rim Light intensity. A value of 0 is the same as no value.

`$rimmask 1` (boolean)

Masks Rim Light with the alpha channel of `$phongexponenttexture`.

Self-Illumination Parameters

`$selfillum 1` (boolean)

Enables Self-Illumination. The alpha channel in `$basetexture` is used as a mask to determine the level of Self-Illumination.

`$selfillumtint "[1 1 1]"` ([R G B])

Tints the color of the Self-Illumination.

`$selfillummask "path/to/selfillum"` (string)

Defines a custom Self-Illumination mask. This mask can contain color information, which will be used to determine the color of the glow. Brighter colors in the texture will glow with more intensity.

Animated Texture Parameters

`Animatedtexture` (Proxy)

Animated textures are loaded with a proxy and allow to read several frames in a VTF file and use them as an animation. The whole syntax is as follows:

```
Proxies
{
    AnimatedTexture
    {
        animatedTextureVar           $basetexture
        animatedTextureFrameNumVar   $frame
        animatedTextureFrameRate     24
    }
}
```

`animatedtexturevar $basetextureuse` (variable/string)

Defines the animated texture that contains the frames. Can be used to define a custom texture or reference `$basetexture` directly.

`animatedtextureframenumvar $frame` (variable/string)

Determines the number of frames in the animation. `$frame` will use all the frames in the texture.

`animatedtextureframerate 24` (integrer)

Speed of the animation in frames per second. Higher values will increase the speed.

Detail Texture Parameters

`$detail "path/to/detail"` (string)

Texture that blends with the albedo to increase detail when looking at a surface from a close distance.

`$detailscale 1` (float)

Scale of the detail texture allowing it to tile and give the impression of higher resolution.

`$detailblendfactor 0.5` (float)

Opacity of the detail texture.

`$detailblendmode 0` (integrer)

Type of blend mode to use when combining with albedo map. This is usually left at 0, which works similar to Photoshop's Overlay blend mode, where grey becomes transparent, while brighter pixels brgiten the image and darker pixels darken it.

Other Parameters

`$model 1` (boolean)

Defines if the object is a world model or not and is needed to make some shaders work on models, like UnlitGeneric or Refract.

`$surfaceprop "metal"` (string)

Defines material type, affecting physical properties, sounds and bullet decals.

Material type names are defined in `<game>\scripts\surfaceproperties_manifest.txt`.

Some common materials are easy to guess like `wood`, `metal`, `concrete`, `plastic`, `grass`, `water`, etc. A list of typical stock Valve material types can be found here.

`$nocull 1` (boolean)

Makes the material double sided, useful when working with transparent material or for elements like hair. Same result can be achieved by manually duplicating and fliping the necessary polygons in the model.

`$translucent 1` (boolean)

Allow the material to be partially transparent. The alpha channel in `$basetexture` is used as a mask to determine the level of transparency, white being opaque and black being completely transparent.

Warning: This parameter may not work when some other parameters are present, such as `$envmap` or `$selfillum`. If `$translucent` doesn't seem to work, `$alphatest` might be useful instead.

`$alphatest 1` (boolean)

Similar to `$translucent` but offers lower fidelity and is faster to draw. It also works better in cases where `$translucent` fails, and handles depth issues better.

Warning: This parameter may not work when some other parameters are present, such as `$envmap`.

`$halflambert 1` (boolean)

Softens the diffuse shading, resulting in lower contrast on the side of the model that's facing away from light. Can be useful in very specific cases, for example when trying to simulate materials like skin or wax.

Parameter Cheat Sheet & Ready to tweak VMT example file

Once you have learned a bit what each parameter does, it's still useful to have them available and organized so that they are easy to look up when you need to. That's why we also made a **VMT Cheat Sheet** with all the commands mentioned in this guide. Check it out in the following link:

Parameter Cheat Sheet

You can also download a **VMT example file** with the most common parameters for skins ready to be tweaked for your model!

VMT Example File

VMT Example File (commented version)

(Use right click and "Save link as..." if you want to download the VMT examples.)

References and sources:

- In-game Testing by Yogensia and Rafaël De Jongh (Left 4 Dead 2, Counter-Strike: Global Offensive)
- Valve Wiki
- PolyCount Wiki