SOLDIER OF FORTUNE II DOUBLE HELIX

SOF2 SHADERS

BY RAVEN SOFTWARE

1.0 Overview

First of all, this document assumes that you are familiar with Quake 3 shaders. If not, please look to www.idsoftware.com, www.telefragged.com, or www.planetquake.com for tutorials and documentation. Although the basics are still pretty much the same as Q3: TA, there have been many modifications to shaders for SoF2.

Second, every texture in SoF2 needs to have a shader entry for material properties such as rock, sand, etc. Third, character models don't retain material properties without a **material map** (material maps are described in SoF2_Character_Setup). Finally, characters need shaders for their **hitmaps** (hitmaps are described in the SoF2_Character_Setup).

This document explains all shader creation via ShaderEd2, a front-end designed specifically for editing shaders. Shaders can be created and modified using a text editor, but Raven strongly suggests the use of ShaderEd2. It will give you instant feedback and takes a lot of the guesswork out of creating shaders.

2.0 Basic Shader Creation

Important note: It is not recommended that original SoF2 shader files be modified. Therefore, start out by creating a new .shader file. Shaders will be stored in base/shaders, not base/scripts as done in Quake 3. To create a new shader file, open ShaderEd2, create some shaders, do a File/Save, and make a name for your new .shader file.

2.1 ShaderEd2 Overview

First, add a group name. A group name is the path to the textures that you will be making shaders for. For example, the group for Colombia textures is 'textures/colombia' and mirrors the path where those textures are stored. To do this, right-click on **base** and **Add Group**.

Once a group name is added, a plus symbol appears next to base. Clicking the plus symbol will expand the tree. If this had been an existing .shader file, you would find one or more sub-groups depending on the shader. Shaders for world architecture (Colombia) generally only have one sub-group while shaders for characters (average_sleeves) might have four or more. Expanding the tree for any sub-group will show you a list of all currently existing shaders. Since you are starting with a new file, this sub-group will be empty.

Right clicking on a sub-group will give you six options:

- **Add Files** - This is the way you add new shaders, click and browse to your texture and a new shader will be created with the exact same name as the texture. Shaders work off of name recognition and are grouped as a whole. Even if you have multiple .shader files the game will treat them all as one group. Be careful not to duplicate names as it will create errors.

- **Sort** – Sort does exactly what it says. You can sort alphabetically, by width, or by height.
- **Filter** – This allows you to filter for different options and is a good way of either comparing shaders or finding a shader. You can filter by dimensions, stages, whether or not it has a damage texture, and whether or not it has an alias shader. The result of the filter will highlight all appropriate shaders in the left hand column.
- **Detail** – This will select and highlight all shaders that have a detail stage on them. Detail stages are stages that are not necessary for the shader to work, and may be culled out on lower end machines or video cards.
- **No Detail** – This will select and highlight all shaders without detail passes on them.
- **Find** – This brings up a dialogue that will allow you to enter text to find a shader. Entering a word that is common to multiple shaders (i.e. rock) will select and highlight all shaders containing the word.

2.2 Adding a new shader

To create a new shader, right-click on **Add Files** and a browser will open. Find the texture you want, select it, and either double-click or click **Open**. This will add a new shader entry with the exact same name. Note that multiple files can be added at one time as well.

Select your new entry from the list on the left, and you will see it displayed at the right in the window. NOTE: When selecting multiple files, this will take a lot of memory from your video card and may slow your machine down.

You will see the name of the shader displayed in gray under a thumbnail of the texture. If you see the name displayed in red, this indicates an error. When a new texture is added though, the name will be gray even though there are no shader properties.

2.3 Giving your shader properties

Double-click on the thumbnail on the right or listed name on the left to open up shader properties. You will find seven tabs across the top and each is described below. Although this document assumes a familiarity with shaders, this section will walk you through setting up a basic SoF2 shader.

- **General** - Gives General information about the shader including width, height, filename, and directory path.
    - **Stretched** - (displays entire texture but distorted to fill screen) Clicking on this button will change the proportions to display the image **1:1** but will crop it to the window.
- **Shader** - This is where you set up a majority of the shader properties. For a basic shader, a material needs to be set from the pulldown in the upper left. The only other thing that might need to be filled in is the **Editorimage** in the upper right corner. If this is left empty and the shader name matches the texture name, the texture will be used for display in Radiant. However, if you duplicate a shader to give special properties, the names will no longer match. Therefore, an

editor image is needed. For all other information on the **Shader** tab, see section 3.0.

- **Stages (Blending)** - This is the page where the blending stages and modes are defined. For a basic shader, all that needs to be done is to click on **Default Setup**. Then, browse to the texture that you are making a shader for and that's it. This will add two stages; a lightmap stage and a map stage. For all other information on the **Stages (Blending)** tab, see section 4.0.
- **Stages (Coords) –** This tab is not needed for a basic shader. This section will allow you to modify texture coordinates on a shader for scaling, stretching, moving, etc. For information on the **Stages (Coords)** tab, see section 5.0.
- **Stages (AnimMap) –** This tab is not needed for a basic shader. This section will allow you to have an 8-frame animated shader. For information on the **Stages (AnimMap)** tab, see section 6.0.
- **Stages (surfaceSprites) –** This tab is not needed for a basic shader. This section allows you to add surface sprites to a shader. For information on surface sprites, see the document SoF2_Vertigons.
- **Tiled –** This tab allows you to see a texture tiled in various ratios, view damage shaders linked to this shader, see the shader animated, with a background if alpha, and more.

3.0 Shader Tab

3.1 Flags section

- **Material** - There are various effects called when interacting with a surface containing a material. Footstep sounds, footstep volume, bullet impacts, bullet penetration, and more are all defined for a material. The number of materials are limited and hard coded. To see what is available for a given material, see base/ext_data/generic.material, which can be found in therest.pk3. Following is a list of the materials provided in SoF2.
    1. None
    2. Solidwood
    3. Hollowood
    4. Solidmetal
    5. Hollowmetal
    6. Shortgrass
    7. Longgrass
    8. Dirt
    9. Sand
    10. Gravel
    11. Glass
    12. Concrete
    13. Marble
    14. Water
    15. Snow
    16. Ice
    17. Flesh
    18. Mud
    19. BPGlass >> this is Bullet Proof Glass
    20. Dryleaves
    21. Greenleaves
    22. Fabric
    23. Canvas
    24. Rock

25. Rubber
26. Plastic
27. Tiles
28. Carpet
29. Plaster
30. ShatterGlass >> Hard coded effect
31. Armor
32. Computer

- **Flags** – Yes we have a flags section in the flags section!
  - **No Picmip** - This forces the shader to have no mipmapping. This should only be used in a few cases, such as User Interfaces and graphics that need resolution to work. It will only use assets from the top level of Picmip (base/textures) Signs that progress game play would be an example of a **No Picmip** shader. This will hit performance hard so use it carefully.
  - **No Mipmaps** - This will force no mipmapping. With this, textures will be in a picmip based on your menu Texture Quality, but there is no real-time downsampling of textures at a radius threshold. This stops textures from having a Moiré pattern at a distance.
  - **No compression** – There is no hardware compression of the texture and should be used for fonts only.
  - **Portal** - Creates mirror effect. (NON-FUNCTIONAL IN SoF2)
  - **Poly Offset** – This will offset the shader from a surface. This is used mainly on decals such as bullet impacts, explosions, posters, etc.
  - **Entity Mergable** - Rendering optimization – alters sorting – rarely if ever used.
  - **Detail +** - Provided here for completeness sake, it is the flag applied to a brush when it is made detail in Radiant.
  - **Alpha Sort** - Helps the engine to sort alpha textures and is applied to most world alpha textures.
- **Content Flags** – These flags relate to properties the brush will have in the world. For instance, if a shader with the water content flag is applied to a brush, the whole brush will become water. Note that mixing content flags between faces on a brush will not work. So, if one face has water, and the other five have fog, the results will be anything from a fog brush to a water brush to the game crashing. Since ShaderEd2 was created to support multiple Raven projects, some flags have no function in SoF2. Also, if a content flag is changed on a shader, a map has to be re-processed to see the effects of the change.
  - **Solid** – Makes a shader solid. This is defaulted on and is checked on every shader unless specifically turned off.
  - **Opaque** – Makes the shader so you can't see through it. This is defaulted on and is checked on every shader unless specifically turned off.
  - **Outside** – Applied to tools/_outside, this will let the weather system know that rain or snow can be drawn. Fill the entire desired area with an outside brush. If this flag is checked, solid should be unchecked.
  - **Lava** – Not used in SoF2.
  - **Water** – Gives a brush water properties and allows the player to swim, drown, and splash. If this flag is checked, solid should be unchecked.
  - **Fog** – Gives fog properties. Brush fog is used in a limited capacity in SoF2, but a fog value placed on the worldspawn is used quite frequently. If this flag is checked, solid should be unchecked.

    - **Player Clip** – A brush with this flag will allow NPCs through, but not the player. If this flag is checked, solid should be unchecked.
    - **Monster Clip** – A brush with this flag will allow the player through, but not any NPCs. If this flag is checked, solid should be unchecked.
    - **Bot Clip** – Not used in SoF2.
    - **Shot Clip** – A brush with this flag will block weapon fire.
    - **Trigger** – This is provided purely for completion sake. The trigger content is set in code when a brush is turned into a trigger entity.
    - **No Drop** – A brush
    - **Terrain** – Used on ARIOCHE terrain via the tools/_roam shader.
    - **Ladder** – Creates a ladder brush so the player can climb. For SoF2 ladders, a ladder brush and a trigger_ladder with a trigger texture must be present in the same place.
    - **Abseil** – A brush with this flag allows NPCs in single player to rappel. Very rarely used.
- **Surface Flags** – These flags change the surface properties of a brush. In theory, these are separated from content flags because they don't require a map to be re-processed if changed. It is always safest to do a re-process to see changes though.
    - **Sky –** Gives a shader the sky properties and opens up the grayed out Sky area to the right.
    - **Slick –** Gives a shader a lower friction and makes the player slide.
    - **No Damage** - No damage shaders will be triggered for this shader.
    - **No Impact** – Bullets and projectiles will be removed on impact, leaving no effect and no marks.
    - **No Marks** – Bullets and projectiles will remain, but triggered effect (i.e. explosion scorch) will not be applied.
    - **No Draw** – Will cause the shader to not draw the face it is applied to. It is used on triggers, caulk brushes, and clip brushes. These brushes will appear in Radiant but not in game.
    - **No Steps** – Footstep sounds will not be played when walking on a surface with this flag.
    - **No Dlight –** Dynamic lights will not be applied to a shader with this flag.
    - **No MiscEnts –** Does not allow MiscEnts (i.e. trees) to be placed in the area. This was used on the Prague truck chase map (pra5) so that trees didn't sprout in the road.
    - **ForceField** – Not used in SoF2.
    - **MetalSteps** – Not used in SoF2.

3.2 Deformations section

Deformations are behaviors applied to a brush that actually changes the physical form of the brush by moving the vertices of the brush. Deformations have not changed from the original Quake 3 code, so they won't be described here. In SoF2, the wave deformation was used to make trees sway in the breeze. The sin deformation was tried for water, but is very expensive due to the way it breaks up the brush and ultimately not used. Finally, the projectionshadow deformation was used on the projectionshadow shader, which will cause models to have a projection-based shadow if applied via the menus. Deformations are fairly expensive, and should be used sparingly.

3.3 Sky section

If the **sky** surface flag is checked, this section opens up. This area is for specifying parameters for skies. Skies can have two layers, although this feature is not used in SoF2. The reason is because our many of our skyboxes have buildings, and the cloud layer could not be masked to prevent drawing over the buildings.

- **Outerbox** – In a two-layered skybox, this would be the base layer (i.e. blue sky). In SoF2, this was the only layer used.
- **Innerbox** – This would allow for a cloud layer to be applied over the Outerbox.
- **Cloudheight** – This designates the height of the Innerbox.
- **Sun Color** – This is the color of the light cast by the skybox. It is listed as an RGB value. Having the sky cast light is a great way to light large areas without lots of point lights.
- **Intensity** – This is how bright the light will be cast from the sky. Values of 80 – 100 are used for dusk or night, while values over 100 are used for daytime.
- **Degrees** – This is the direction of the light source or sun. For example, if 315 were the **degrees**, light would be casting from the direction of 315 if viewed in Radiant.
- **Elevation** – While **degrees** is basically the yaw value of the sun, **elevation** determines the pitch. A value of 0 places the sun at the horizon, while a value of 90 places the sun straight overhead as if it is noon. For dramatic shadows, values around 30 to 40 work best.

3.4 Sort section

Sorting of shaders can sometimes be a tricky business for the game to figure out, so adding a sorting helper can alleviate the problem. This is defaulted to off for all shaders, and should only be used by advanced users. In SoF2, these were used very rarely, and never on world shaders. They are mostly needed for decals and bolt-ons for Ghoul 2 models.

- **Value** – Not used in SoF2.
- **Enum** – There are 14 different values in this pulldown. The first 9 listed are original Quake 3 values and their descriptions can be found in documents pertaining to Quake 3 shaders. The remaining 5 were created for use with sorting issues on Ghoul 2 models.
    - **portal**
    - **sky**
    - **opaque**
    - **decal**
    - **seeThrough**
    - **banner**
    - **additive**
    - **nearest**
    - **underwater**
    - **inside** – Force sorting to draw this shader last.
    - **mid_inside** – Force sorting to draw this shader before all other shaders except those marked **inside**.
    - **middle** – Force sorting to draw this shader before all other shaders except those marked **inside** and **mid_inside**.

  - **mid_outside** – Force sorting to draw this shader before shaders marked **middle**.
  - **outside** – Force sorting to draw this shader first.

3.5 Fog section

If a shader is flagged with the content flag **Fog**, this section is opened up. There is a color chooser and RGB entry boxes for choosing the color of the fog. There is also a field called **OpacityDepth** that specifies the distance at which the fog becomes opaque. When used for worldspawn fog, it is best to make the **OpacityDepth** slightly smaller than the distancecull placed on the world.

3.6 Misc section

This area is for things that don't fit anywhere else. They are described individually below.

- **Cull type** – By default, any given polygon is one-sided. Changing this value can cause a polygon to draw the texture on the backside instead, or on both sides. Cull disable is generally used on items made from patch meshes such as banners or on flat-polyed Ghoul models such as the collars on shirts.
- **Damage Shader** – Allows you to browse and link to a damage shader. This will be triggered when the health of this shader is exceeded. The health is specified in the **Damage** entry to the right
- **Alias Shader** - Alternate simple shader for increased performance on low-end machines. Generally, you will want the alias shader to be similar and congruent with the area it will be used in. For instance, if there are three signs that are all the same size and have different text, but are really just fluff, making two of the signs have an alias shader of the first will cause all three to appear the same if alias shaders is turned on in the menu.
- **Hit Location** – This is a character model related field allowing you to link this Shader to a Hit location map. Hit location maps are described in detail in the SOF2_Character_Setup document.
- **Hit Material** – This is a model specific field that allows you to link material maps to a shader. While world architecture simply needs a material applied, any model in the game that can be interacted with (i.e. Ghoul models) needs a material map. Basically, you duplicate your original texture (i.e. an NPC's torso), then paint over it in solid colors to define materials. This way, one model can have blood over flesh areas, metal over armor and bet buckles, etc. Material maps are defined in the SoF2_Materials_Index_Keys document.

3.7 QERadiant section

This section defines shader properties solely for Radiant and the compile tools; the game itself doesn't care about these entries.

- **EditorImage** – Sometimes, an image is used in multiple shaders. By default, Radiant only shows an image if the texture name and shader name matches. Specifying an **EditorImage** allows the shader to show up in Radiant.
- **No Carve** – This is for tool textures that are used frequently, but should not be part of the bsp. Triggers and clip brushes have this flag.

- **Trans** – Will cause the texture to be transparent in Radiant (but not in game). It is mainly used for tools so that the designer can see what they are doing.

3.8 Q3Map section

Items located in this section make fundamental changes to brushes that require a map to be processed when a shader with one of these properties is added. Brushes that cast light, brushes that are vertex lit, or brushes that cast special shadows (due to alpha textures for instance) are all part of this section. Most of these are unchanged from Quake 3 and are not described here.

- **Lightimage** – Instead of specifying an RGB color, you can specify an image to have a shader cast light. It uses the most predominant color in the image for the light color.
- **Light Color** – This RGB value can be specified to cause a shader to cast light.
- **Flare** – This allows a shader to have a lens flare effect if the feature is turned on in the menus. It is used for lights and only needs to be a texture (not another shader).
- **Backsplash Fraction**
- **Backsplash Distance**
- **Surface Light**
- **Light Subdivision**
- **Tesselation Size**
- **Vertex Scale**
- **Lightmap Sample Size**
- **Hint**
- **No Lightmap**
- **Alpha Shadow**
- **Global Texture**
- **Trace Light**
- **Patch Shadow**
- **No T Junctions**
- **Area Portal+**
- **ClusterPortal+**
- **Origin+**
- **Structural+**
- **Vertex Light**
- **Visible Light**
- **Vertex Shadow**

4.0 Stages (Blending)

4.1 Stage section

When a new shader is made, only three buttons are active in this section. Clicking on any of these three buttons can create a shader stage. The **New** button allows you to add each stage manually. The **Default Setup** button creates a two-stage lightmap shader for world architecture. The **Hard Edged Alpha** button creates a three-stage alpha shader for world architecture.

A default world shader in a text editor looks as such:

```
textures/colombia/bamboo_roof3
{
    q3map_material HollowWood
    cull disable
  {
    map $lightmap
  }
  {
    map textures/colombia/bamboo_roof3
    blendFunc GL_DST_COLOR GL_ZERO
  }
}
```

When the **Default Setup** button is pressed, the two stages above are added. In ShaderEd2, the lightmap stage will have the default values in the four pulldowns in the **Map** section. These are:

- Src Blend –none-
- Dst Blend –none-
- Depth Func lequal
- Alpha Func –none-

The stage with the actual image has the following:

- Src Blend GL_DST_COLOR
- Dst Blen GL_ZERO
- Depth Func lequal
- Alpha Func –none-

Note this is different than base Quake 3, which used GL_DST_COLOR/GL_SRC_COLOR.

Now for the rest of the Stage section.

The pulldown list will allow you to see a list of all stages and choose the desired one. The two arrows to the right of the pulldown list will allow you to move up and down the list of stages without using the pulldown.

- **New** - This button allows you to add new stages.
- **Dup –** This button allows you to duplicate the currently selected stage.
- **Delete** – This button allows you to delete the currently selected stage and will bring up a confirmation dialogue.
- **Up and down arrows –** These arrows to the right of the previous buttons allow you to move stages up and down in the list.

The pulldown list to the right that defaults to **Stretch** allows you to change the proportions of the image only for display purposes.

- **Animate** – This button will display the animated frames of the shader or anything else that modifies the shader (AlphaGen, RGBGen, TcMod, etc).
- **Draw Detail –** This button is defaulted on so that all stages are shown. If turned off, stages marked detail will not be shown. If detail stages are added, this is a good way to ensure that the shader still looks good without them.

- **Stage = end** – This will only show stages up to the stage currently on. So, if a four-stage shader is used, and you move to stage 3 and turn this on, the shader will show how it looks without stage 4.
- **Error Check** – When any potential error occurs in a shader, this button will blink on and off with the number of errors in the shader. When used it will display the errors in detail. However, some shader may always show up invalid even though they are not (Arioche shaders for example).
- **Lightmap = Full** – When a lightmap stage is included, ShaderEd2 uses a gradient at the lightmap level to show what the image will look like in full light and no light. Using this button will show the texture in full light.
- **Background** – This will display a multicolored background behind the shader and is useful for checking if alpha channels are working correctly in a stage.

Above, a standard lightmap shader was covered. Here, a standard **Hard Edged Alpha** and **Soft Edged Alpha** shader will be covered. Note that the **Soft Edged Alpha** is grayed out and cannot be selected initially. This button can be pressed after either of the other defaults is made, but it will generally cause a stage error. As mentioned above, this does not necessarily mean the shader is wrong. ShaderEd2 expects a lightmap stage to be followed by a standard map stage, which is why the error occurs. Following is a standard **Hard Edged Alpha** as seen in a text editor. A **Hard Edged Alpha** is defined as the alpha channel values being 0,0,0 and 255,255,255 only (TRANSPARENT).

```
textures/common/chainlinkfence_top
{
    surfaceparm        nonsolid
    surfaceparm        nonopaque
    surfaceparm        playerclip
    surfaceparm        monsterclip
    q3map_material     SolidMetal
    q3map_alphashadow
    cull        disable
  {
     map textures/common/chainlinkfence_top
     alphaFunc GE128
     blendFunc GL_SRC_ALPHA GL_ZERO
     depthWrite
  }
  {
     map $lightmap
     blendFunc GL_ONE GL_ZERO
     depthFunc equal
  }
  {
     map textures/common/chainlinkfence_top
     blendFunc GL_DST_COLOR GL_ZERO
     depthFunc equal
  }
}
```

In ShaderEd2, the first stage is a map stage with the four pulldowns set up as follows:

- Src Blend – GL_SRC_ALPHA
- Dst Blend – GL_ZERO
- Depth Func – lequal
- Alpha Func – GE128

Also, the **DepthWrite** field is checked in the first stage. The second stage is a lightmap stage with the following setup:

- Src Blend – GL_ONE
- Dst Blend – GL_ZERO
- Depth Func – equal
- Alpha Func – none

Finally, another map stage is added as follows:

- Src Blend – GL_DST_COLOR
- Dst Blend – GL_ZERO
- Depth Func – equal
- Alpha Func – none

A **Soft Edged Alpha** is a shader with the alpha channels values set *between* 0,0,0 and 255,255,255 (TRANSLUCENT). A standard **Soft Edged Alpha** as seen in a text editor follows:

```
textures/common/glass_d
{
        surfaceparm      nonopaque
        q3map_material   ShatterGlass
        q3map_nolightmap
        q3map_onlyvertexlighting
        cull        disable
  {
     map textures/common/glass_d
     blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
     alphaGen vertex
  }
}
```

Note that in SoF2, **Soft Edged Alpha** shaders are vertex lit and have no lightmap. Therefore, they generally only have one stage. This stage is setup in ShaderEd2 as follows:

- Src Blend – GL_SRC_ALPHA
- Dst Blend – GL_ONE_MINUS_SRC_ALPHA
- Depth Func – lequal
- Alpha Func – none

4.2 Map section

The following are radio buttons, and only one can be selected per stage.

- **Map** – A standard image and the workhorse selection.
- **ClampMap** – A standard map, but clamped so it won't tile. Used a lot for sprites (i.e. muzzleflashes).
- **Lightmap** – Default standard lightmap and another workhorse selection.
- **Whiteimage**—Procedurally created pure white image.
- **VideoMap** – Not used or functional in SoF2.
- **AnimMap** – Multi-frame animated shader that works in conjunction with the **Stages(AnimMap)** section of ShaderEd2.
- **ClampAnimMap** – Multi-frame animated shader that is also clamped so it won't tile. This works in conjunction with the **Stages(AnimMap)** section of ShaderEd2.

- **AnimMap (one shot)** – A non-looping animated shader that works in conjunction with the **Stages(AnimMap)** section of ShaderEd2.

The rest of the map section follows:

- **Detail** – This checkbox will make this stage detail, and cull it out on low-end machines.
- **Depthwrite** – Adds the depthwrite field for some alpha textures.
- **Image field** – This is the path to the image used for this stage. The button at right allows you to browse to the image.
- **Blend Picker** – Clicking on this button will take you to a sample screen. This sample screen will display a green box with multiple thumbnails. These thumbnails are various blend modes that can be used. Anything in the green box is considered a valid blend mode; anything outside this green area is unsupported for SOF2. By moving your cursor over the thumbnails, the blend modes are displayed in text form at the bottom right. This is a very useful tool to get immediate feedback on determining the correct blend mode to use. This page also offers some of the same functionality as the Blending Stage Page and function the same.

4.3 RGBGen section

RGBGen functionality is unchanged from Quake 3 and will not be described here. However, **LightingDiffuse** is the RGBGen used for all models in SoF2.

4.4 AlphaGen section

AlphaGen functionality is unchanged from Quake 3 and will not be described here.

4.5 TCGen section

TCGen functionality is unchanged from Quake 3 and will not be described here.

5.0 Stages(Coords)

The top portion of this section retains the functionality of the **Stages(Blending)** area. The only thing to note is that to modify a stage, it must selected in the pulldown area.

5.1 TCMod section

This area lets you modify the textures coordinates for a shader by scrolling, rotating, etc. Each stage can have up to four TCMods assigned to it. TCMod functionality is unchanged from Quake 3 and will not be described here.

6.0 Stages(AnimMap)

The top portion of this section retains the functionality of the **Stages(Blending)** area. The only thing to note is that to modify a stage, it must selected in the pulldown area.

The Quake 3 engine limits the number of frames for an animMap to 8. Additional frames are simply dropped.

To create an animMap, the **AnimMap**, **AnimMap(Clamp)**, or **AnimMap (one-shot)** radio button must be selected. Otherwise, this window is grayed out. Next, click on the browser button towards the bottom. Add one image at a time in the order you want them played by clicking **Add** or **Insert** once you have an image in the browser. You can add an image more than once. You can delete frames, reorder them, insert new frames, etc. with the controls above the browser. Enter a number in the **FPS** field to specify how fast the frames will play.

Once you have everything set up as desired, you MUST hit the **Re-evaluate frames** button. Then, you can view the results at right. If anything is changed, you must hit this button again or the changes won't be saved.