TERRAIN CREATION AND TEXTURING

BY RAVEN SOFTWARE

1.0 Introduction

There are two types of terrain in Soldier of Fortune II, each with its own method for creation and texturing. One uses terrain created in GenSurf, a program freely available for download at http://tarot.telefragged.com/gensurf/ . Basically what this does it take a bitmap that you create in a paint program, and applies it as a heightmap, carving hills and valleys. With this method, textures are applied via metashaders, an extended part of the shader system. We are using the same method as that used for Quake 3: Team Arena, so if you are familiar with terrain creation for that, you'll know how to do it for SoFII.

The second method uses an algorithm to create an ARIOCHE terrain. The method is fairly similar to GenSurf in that a heightmap is used to create terrain. However, with ARIOCHE, the engine does the generation on map loadtime. This allows for fast editing (all you have to do is modify the bitmap), and much larger terrains. The Random Mission Generator uses ARIOCHE to generate its terrain for both single and multiplayer games.

2.0 GenSurf Terrain

2.1 Creation

We won't get into the specifics of creating a GenSurf terrain here. The GenSurf website has documentation and tutorials on that. There are only two SoFII specific things you need to know about using GenSurf. One is that under texture/surface, you'll want to point it to tools/_terrain. Second, under textures/other, you'll want to point it to tools/_caulk. Notice that the underscore in the path is part of the SoFII tool shader naming convention.

Take the image that you made the Gensurf terrain with and save a copy into base/textures/heightmaps. This will be the image read by the game to apply textures to the terrain.

Open that image in a paint program and 'posterize' it to how many layers of texture you want in your terrain. Generally, (255,255,255) white is used for the top layer, (0,0,0) black for the bottom, and (128,128,128) gray for the middle. If you are going to have more layers, even divide the gray colors (don't use 128 gray and 129 gray for example).

This image, the **alphamap**, has a few important qualities. First, it has an alpha channel that is exactly the same as the RGB, thus the term alphamap. Second, its size is the same as the divisions in the Gensurf terrain. Thus, if the terrain has 32 by 32 divisions, the image is 32 by 32 pixels. Note that the image has to be a power of 2 regardless of the divisions (16x16, 32x32, 64x32, etc). Third, you can never have an area where three colors meet. If using three colors, always surround the high and low color with the middle one so the high and low don't touch. Having three colors touching will result in a bad texture blend. Finally, the image must be rotated 180 degrees due to an error in our code. If it isn't rotated, you won't get the desired results.

NOTE: There is another version of bitmap you can use, but it is untested in production maps. You can make an 8-bit PNG with no alpha channel. This PNG does not have to be rotated and can be any size.

2.2 Radiant

Now that you have a Gensurf terrain and an alphamap, you need to set up the terrain in the level editor, Radiant. This document assumes you are familiar getting around Radiant and creating maps.

In Radiant, make sure your terrain is assembled as a **func_group**. Make sure it has the **tools/_terrain** texture on the visible surfaces and **tools/_caulk** on the non-visible ones. For any given brush in the **func_group**, only one face should have the _terrain texture on it. In the **func_group** parameters, have the following:

*"**terrain**"* *"1"*
Make the group into terrain

*"**alphamap**"* *ex: "textures/heightmaps/col3"*
The image that determines where textures are applied

*"**layers**"* *"3"*
How many textures you want on the terrain. 3 gives good variety without being too complicated

*"**shader**"* *ex: "metashader/col3"*
The name of the shader (see below)

Note: If you do a region BSP of the terrain, the results won't be as expected as the alphamap will only be applied to the region. To get the proper results, you must process the whole map or a region encompassing the entire terrain group.

2.3 Metashaders

This document assumes basic knowledge of shaders. Metashaders are more easily done through a text editor (Notepad) instead of ShaderEd2 as it's easier to cut and paste stuff that you need to repeat.

For a metashader using three textures, you need six shaders. The reason? There are three shaders for the textures themselves, and three shaders for the blend between the textures. Here's an example of a 3-layer metashader:

textures/metashader/col3_0
textures/metashader/col3_1
textures/metashader/col3_2
textures/metashader/col3_0to1
textures/metashader/col3_0to2
textures/metashader/col3_1to2

Notice that the beginning of all of these matches up with the “shader” key value on the terrain group. The _# signifies the level at which the texture will draw. The “0” texture will draw on the black area of the alphamap, the “1” on the gray area, and the “2” on the white area. The remaining three shaders blend between the three textures. Note that if you have a “0to1”, you don’t need a “1to0”.

Following is an example of how a metashader named ‘col3’ is set up.

All textures have the flags, “**q3map_nolightmap**”, “**q3map_onlyvertexlighting**”, and “**q3map_vertexshadows**” set on them (see the shader document for details).

The three base textures have only one pass with the texture, **rgbGen vertex**, and a **tcMod** scale set. Note that **tcMod** scale is the inverse of scaling in Radiant. When scaling a texture in Radiant, a fractional value makes the texture smaller, or tile more on a face. In shaders, a smaller number actually makes it larger. The reason is that the shader reads as, “Put .5 of this texture on this face” rather than scale the texture to .5 of its size.”

The three bridge textures have two blend passes. Both passes have **rgbGen vertex**, **alphaGen vertex**, and **tcMod scale**. The second pass also has **blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA**. This is so the two textures actually blend.

```
// ************************************************
// *   COL3 Metashader
// ************************************************

textures/metashader/col3_0
{
        q3map_nolightmap
        q3map_onlyvertexlighting
        q3map_vertexshadows
   {
        map textures/colombia/mudside_b
        rgbGen vertex
        tcMod scale .25 .25
   }
}
textures/metashader/col3_1
{
        q3map_nolightmap
        q3map_onlyvertexlighting
        q3map_vertexshadows
   {
        map textures/common/grass
        rgbGen vertex
        tcMod scale .25 .25
   }
}
textures/metashader/col3_2
{
        q3map_nolightmap
        q3map_onlyvertexlighting
        q3map_vertexshadows
   {
        map textures/colombia/rock02
        rgbGen vertex
```

```
		tcMod scale .25 .25
	}
}
textures/metashader/col3_0to1
{
		q3map_nolightmap
		q3map_onlyvertexlighting
		q3map_vertexshadows
	{
		map textures/colombia/mudside_b
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
	{
		map textures/common/grass
		blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
}
textures/metashader/col3_0to2
{
		q3map_nolightmap
		q3map_onlyvertexlighting
		q3map_vertexshadows
	{
		map textures/colombia/mudside_b
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
	{
		map textures/colombia/rock02
		blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
}
textures/metashader/col3_1to2
{
		q3map_nolightmap
		q3map_onlyvertexlighting
		q3map_vertexshadows
	{
		map textures/common/grass
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
	{
		map textures/colombia/rock02
		blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
		rgbGen vertex
		alphaGen vertex
		tcMod scale .25 .25
	}
}
```

3.0 ARIOCHE Terrain

3.1 Creation (basic)

There are two parts to creating terrain for an ARIOCHE map, and one step is very similar to GenSurf. First, create a bitmap that will act as the heightmap. Use a grayscale image, but have as many colors as necessary. The image must still be a power of 2 (i.e. 32x32, 128x128) just like with GenSurf, and must be 8-bit. Save the heightmap in base/heightmaps (not base/textures/heightmaps like GenSurf).

Next, create a .terrain file in base/ext_data. The .terrain file determines the altitudes at which a texture should be placed. There is no need to create blend shaders as the terrain does this automatically. Below is an example of a .terrain file. Text in blue is not in the file.

```
col9                                                    name of the map and the terrain file
{
        altitudetexture                                 designates a terrain texture
        {
                height 140                              designates the height to start at (140 to 255)
                shader textures/colombia/grass_col9     designates the shader to use
        }
        altitudetexture
        {
                height 120
                shader textures/colombia/floor_moss_col9
        }
        altitudetexture
        {
                height 50
                shader textures/colombia/dirt_terrain1
        }
        altitudetexture
        {
                height 0
                shader textures/colombia/mudside_col9
        }
        water                                           designates a water texture (only 1)
        {
                height 80                               designates the top height of the water
                shader textures/colombia/water_col9b    designates the shader to use
        }
}
```

When a height is listed, it is the height a texture starts at. The height limit is 255. In the above example, mudside_col9 will be applied to the terrain from an elevation of 0 to 49. At elevation 50, dirt_terrain1 will be applied. However, the terrain is blended as well, and there will not be a sharp line between elevations.

3.2 Radiant

In Radiant, create a brush as big as you want the terrain to be, texture it with tools/_roam, and turn it into a **terrain** entity. In the **terrain** parameters, have the following:

*"**heightmap**"* *ex: "heightmaps/col9"*
The image that determines the hills and valleys in the terrain

*"**terraindef**"* *ex: "col9"*
The name of the .terrain file in base/ext_data

*"**numpatches**"* *ex: "650"*
How many patches the terrain is broken into (see below)

*"**terxels**"* *ex: "4"*
The number of terxels each patch contains (see below)

*"**texturescale**"* *ex: ".005"*
The scale of a texture on a given face (see below)

Optional values:

*"**densitymap**"* *ex: "heightmaps/col9density"*
The image that determines the density of miscents on the terrain

*"**miscentdef**"* *ex: "col9"*
The name of the .miscent file in base/ext_data

3.3 Shaders

Shaders for ARIOCHE terrain must be vertex lit, just like that for other terrain. Below is an example of a shader used in the Random Mission Generator. Text in blue is not in the file.

```
textures/arioche/jungle_1
{
        qer_editorimage textures/colombia/mudside_b      an editor image is given if different from the
                                                         shader name
        q3map_material   Mud
        q3map_nolightmap
        q3map_onlyvertexlighting
  {
    map textures/colombia/mudside_b
    rgbGen exactVertex                                   will light the terrain the same regardless of
                                                         r_overbrightbits. A 'vertex' value is also
                                                         possible
    tcMod scale 0.4 0.4                                  a scale value is optional
  }
}
```

3.4 Miscents

Miscents are the models that are placed on the terrain to make it appear more like real terrain. Rocks, trees, and plants are all models that have been used as miscents. One important thing to note about miscents is that they have no collision. It is better to place them out of reach of the player to keep believability intact.

The model format is md3. SoFII has a variety of trees already set up, but if you create your own, make sure to include LODs. This document does not explain how to create md3s or LODs though.

Above, there were two optional fields mentioned for the terrain entity, **miscentdef** and **densitymap**. The **densitymap** is a grayscale bitmap that determines the number of miscents placed in any given area. The image is 8-bit grayscale with black representing areas of low density and white, higher density. The **miscentdef** value refers to a file with a .miscent extension that resides in base/ext_data. Below is an example of a .miscent file. Text in blue is not in the file.

```
col9                                                    name of the miscent file
{
        miscent                                         determines the start of a miscent
        {
                height 130                              the minimum height a model can start appearing
                maxheight 180                           the maximum height a model can appear
                model
                {
                        name models\objects\colombia\jungle\tree08      path to the model
                        frequency 18                    an arbitrary number that determines how often a
                                                        model shows up in a given density area
                        minscale .7                     to give variance, the model can be scaled down
                        maxscale 2.0                    or up by a percentage of its original size
                }
        }
        miscent
        {
                height 130                              if no maxheight is given, it will go to the 255 maximum
                model
                {
                        name models\objects\colombia\jungle\tree09
                        frequency 12
                        minscale .4
                        maxscale 1.3
                }
        }
}
```

3.5 Creation (advanced)

This section explains the relationship between **patches**, **terxels**, and **texturescale**. Patches and terxels together determine how many triangles make up the terrain. The terrain is first divided into a number of square patches. These patches break the terrain into regions that are used for distance culling. Culling is very important due to the size of ARIOCHE maps.

Example 1: With a map that is 4000 x 4000 in the x,y plane, and **patches** set to 16, the terrain will be split into 4 patches x 4 patches and each patch will be 1000 units.

Example 2: With a map that is 8000 x 2000 in the x,y plane, and **patches** set to 16, the terrain will be split into 8 patches x 2 patches and each patch will be 1000 units

Each patch is next divided into smaller segments for rendering. The number of **terxels** defines how many subdivisions there are down each edge. Note that while **patches**

value denotes a total number, the **terxels** value denotes how many per side on a given patch. Valid numbers for this are 2,4, or 8. Other numbers between 2 and 8 will work, but physics on the resulting terrain will have problems. Only use other numbers if physics aren't needed (like in the col9 map).

Example 1: With either of the above examples, you have a patch of 1000 x 1000. A value of 5 **terxels** would give 25 quads (50 triangles). Each triangle would be 40 x 40 units. Since there are 16 **patches** of 50 triangles, the terrain is made up of 800 triangles.

Coming up with the **texturescale** value is a bit tricky. Basically, you just want a value that will make your textures look good. With a 1024x1024 texture, we tended to use a value that would get the texture close to applying to 256 units. Here's the equation to come up with that:

((size of map side) / (sqrt(patches) * terxels)) / (6400)

The numerator is the value you want close to 256, so you need to modify your map size, number of patches, or terxel value to get that.

Example 1: Using our previous numbers, the size of our map is 4000 x 4000, our **patches** value is 16, and our **terxels** value is 5. So:

((4000)/(sqrt(16) * 5)) / 6400
(4000/20) / 6400
200 / 6400
.03125 (this is your **texturescale** value)

Notice that a value of 200 is a bit small, but acceptable. To get closer to the 256 value, decrease **patches** or decrease the **terxels**.

Changing **patches** to 9 gives us:

((4000)/sqrt(9) * 5)) = 266.67

With this solution, you would decrease your triangle count from 800 to 450. That may produce a terrain that is too coarse for your needs.

Changing **terxels** to 4 gives us:

((4000)/sqrt(16) * 4)) = 250

With this solution, you would decrease your triangle count from 800 to 512. That is still less triangles, but greater than the previous solution.

Note that the examples above are theoretical too. A terrain of 4000 x 4000 would probably be too small to make a good ARIOCHE terrain, and would be much better off with GenSurf. By way of comparison, our single player Colombia: Helicopter Extraction (col9) map is 80832 x 80832.