Making Atlas2 maps for TGEA

Author Madcowthomas
Date 30th of May 2007

This is a collection of data that I have been collecting since I start trying to learn this TGEA pipeline. I'd like to say this is information from several people that help me during this process.

Please let me know if something is misspelled or incorrect and I'll correct it ASAP.

Thanks to everyone that helped me and thanks in advanced for any help in the future. This is collection of data and not be regard as set instructions.

Part 0: Notes on Scripting in TGEA

Please note at this time the L3DT Torque exporter only works with files 2048 and under. It is therefore recommended to run the commands through the console in TGEA.

To run a script in the console, type the following:

```
exec ("FOLDER/FILENAME.cs")
```

It is recommend that you run all commands for each part of the process (there are four) in separate script files, as this lets you see the errors or any problems that might pop up. I myself have the four scripts named out as part1.cs and so forth, this allows me to up arrow and change the number only to call the next script.

Part 1 of this scripting process is the most important, this script you have run over a few times till your able get the numbers down to the correct parameters.

Part 1: Converting the .raw file into a geometry chunk file

Below is the basic file you will run to convert the .raw file to a geometry chunk file. You will need to substitute the values below with your values.

```
atlasOldGenerateChunkFileFromRaw16(
"[data/terrains/heightmap.raw",
1024,
1.0,
0.0018,
"data/terrains/deleteme.geometry.chu",
1,
2);
```

The variables are:

```
atlasOldGenerateChunkFileFromRaw16(
"YOUR_MOD_FOLDER/data/terrains/heightmap.raw",
HEIGHTMAP_SIZE,
METERS_PER_PIXEL,
HEIGHT_SCALE,
"YOUR_MOD_FOLDER/data/terrains/deleteme.geometry.chu",
HEIGHT_ERROR,
TREE_DEPTH );
```

HEIGHTMAP_SIZE

This is the side-length of the heightfield, which must be square and a power of two. The actual side-length of the heightfield file must be ONE MORE THAN THIS. Here are some examples:

HEIGHTMAP_SIZE	Size of file
512	513×513
1024	1025×1025
2048	2049×2049
4096	4097×4097
8192	8193×8193
16386	16387×16387
32768	32769×32769

L3DT does not make maps in these power-of-two-plus-one sizes. Instead, make a power-of-two map (e.g. 1024×1024), and then use the 'resize for export' option in the export wizard to change the size in the file.

Information on your .raw files

It's a raw file, so right click on your .RAW file and select 'Properties'. The 'Size:' should be 2,101,250 bytes which is 1025 x 1025 x 2.

If the size is 1,050,625 then it is 1025 x 1025 (the right size) but only 8 bits. If the size is 2,097,152 then it is 1024 x 1024 (the wrong size) but 16 bits(right bit size). If the size is 1,048,576 then it is 1024 x 1024 (the wrong size) and only 8 bits.

METRES_PER_PIXEL

This is the horizontal spacing between points in your heightmap, measured in metres. For Torque it is recommended to use values of 1 or 2 metres per pixel. Consider that a 1024 heightmap at one meter per pixel is 1 sq km (or a little under 2/3 of a mile square). It may just take a few imports and play tests before you decide. Remember that this value can also be fractional.

Getting the meters per pixel from L3DT

This value was set when created the design map, in the "Heightfield resolution (m)" field. You can retrieve this value by using the '*Operations→Heightfield→Change horiz. scale*' menu option.

Getting the meters per pixel from Freeworld3D

Freeworld3D provides these values for you in the world.cfg:

```
<Terrain Size="1024" Step="1.000000">
<Heightmap File="/terrain/heightmap16bit.raw" Format="16bit RAW"
Scale="0.002777"/>
</Terrain>
```

The Step="1.00000" bit is the METERS_PER_PIXEL value you write in the TGEA script file.

HEIGHT_SCALE

The RAW file stores the height values as integers between 0 and 65535. To convert these integer values back to real-world units in Atlas, a scaling factor is used. This value is the total height range divided by 65536. For example, if your terrain has a vertical scale of 1000 (minimum height 300 and maximum 1300), then pass in "1000/65536" for that parameter. Torque will evaluate the math equation when it performs the function. To make your terrain features more dramatic, increase the height range (number before the '/'). These are usually big jumps, so bump it up a few hundred at a time.

Getting height scale from L3DT

To get the height scale from L3DT, go to '*Operations→Heightfield→Change vert. scale*' in the menu. In that dialog there will be an 'alt range' field. That number is what you write in the TGEA script as "[AltRange]/65536" (e.g. "1000/65536").

Getting the height scale from Freeworld3D

Freeworld3D provides these values for you in the world.cfg:

```
<Terrain Size="1024" Step="1.000000">
<Heightmap File="/terrain/heightmap16bit.raw" Format="16bit RAW"
Scale="0.002777"/>
</Terrain>
```

The Scale="0.002777" bit is the HEIGHT_SCALE you write in the TGEA script file (you don't need the /65536 bit; FreeWorld3D calculates that for you).

HEIGHT_ERROR

This value is used to decide how much error - error being deviation from the heightmap - to allow in the mesh generated from the heightmap. Too large an error tolerance and you will get significant visual popping of LODs and the mesh won't follow the heightmap closely. Too small and your performance is terrible and your final atlas file will be huge.

For the numbers for the height error is best always to start with a high number. For example for a 4096 map size start off with a number 20, you end up working this number down.

Example of a map work on other night, I started with 20 and end up with 0.5 when I was done.

Here is a break down of reducing the numbers in half:

```
20, 10, 5, ,1, 0.5, 0.25
```

TREE_DEPTH

Tree depth is used to balance the number of vertices per LOD mesh in the terrain. It is recommended that you have between 1000 and 32000 vertices per level, but you will have to play with this to get good results. Atlas2 will give you some suggestions, but mostly HEIGHT_SCALE, HEIGHT_ERROR and TREE_DEPTH are adjusted through trial and error.

Recommended values for tree-depth are:

Map size	Tree depth
1024×1024	2
2048x	3

4096x	4
8192x	5
16384x	6
32768x	7

When you export you'll see a table like this one:

```
============ Chunk Statistics ============
Level Count Avg. Size
0 256 761.562500
1 64 2978.140625
2 16 8176.562500
3 4 22972.250000
4 1 73725.000000
=========================================
chunks: 341
input verts: 4194304
output verts: 58817
avg verts/chunk: 172
output bytes: 12645
bytes/input vert: 0.00
bytes/output vert: 0.21
real triangles: 682000
=========================================
```

In this example we specified a tree depth of 4. You can see the Avg. Size (actually average triangles) in each terrain LOD level. Ideally you want more than 1000 vertices per level (since the triangles are rendered as strips you can estimate verts = triangle count * 2), but it isn't mandatory. Also you want to keep each level under 32K vertices. Many times this isn't possible with the top levels of LOD like in this example without increasing the error tolerance. As long as you never get far enough away from the terrain to show the top level LOD block this doesn't matter.

So basically this is a trial and error process. I suggest starting with a value of 4 to 6 and doing a few imports before you settle on one value. If you wanted to be really thorough you could go as far as measuring frame rate with different tree levels, but in general this isn't necessary.

Part 2: Generating Atlas2 Geometry

Now we need to generate the actual Atlas2 geometry. Enter the following command in your console window.

```
importOldAtlasCHU(
"data/terrains/deleteme.geometry.chu",
"data/terrains/geometry.atlas");
```

Again, include the full path to your chunk and Atlas geometry files. The variables are pretty self-explanatory here:

```
importOldAtlasCHU(
"YOUR_MOD_FOLDER/data/terrains/deleteme.geometry.chu",
"YOUR_MOD_FOLDER/data/terrains/deleteme.geometry.atlas" )
```

Part3: Generating Atlas2 Textures

Now we will generate our Atlas2 texture map. Enter the following command in your console

window:

```
atlasGenerateTextureTOCFromLargeJPEG(
"data/terrains/colormap.jpg",
2,
"data/terrains/texture.atlas");
```

As always, include full paths. The middle number, 2 in the example above, is the same TREE_DEPTH as used in Part 1.

Generating Atlas2 Textures From Mosaic Maps

To generate an atlas texture file from an L3DT mosaic map, use the following function:

```
atlasGenerateTextureTOCFromTiles(leafCount, tileMask, outFile, outFormat)
```

The argument are:

leafCount	The size of the grid of tiles on a side.
tileMask	The path for the tiles (no extension) with %d for x and y, e.g.,'YOUR_MOD_FOLDER/data/terrains/colormap_x%dy%d'. No extension is needed.
outFile	The file to output a new .atlas file to.
tileFormat	Input file format. If you are using jpeg use "0" while if you are using png use "1".

The "%d" in the *tileMask* argument are placeholder for the script, which are replaced by the x and y coordinates of the tiles when loading. Note that L3DT always uses the "mapname_x%dy%d" naming scheme for mosaic tiles.

Here is an example:

```
atlasGenerateTextureTOCFromTiles(   4,
                                        "YOUR_MOD_FOLDER/data/terrains/colormap_y
%dx%d",
                                        "YOUR_MOD_FOLDER/data/terrains/deleteme.l
ightmap.atlas",
                                        0 );
```

Part4: Putting it all together

At last, we can generate our final Atlas2 terrain! We are generating a "unique terrain" which uses a single texture file overlaid on the terrain. This tends to be the most accurate type of textured terrain, but it usually takes up very large amounts of disk space. A "blended terrain" is somewhat more efficient space-wise, but that will be the subject of another tutorial...

Enter the following command in your console window:

```
atlasGenerateUniqueTerrain(
"data/terrains/terrain.atlas",
"data/terrains/geometry.atlas",
"data/terrains/texture.atlas");
```

At this point you are done creating your files needed for Torque. The next step is too remove the unwanted files and edit your '.mis' file to tell it the file name for the '.atlas' file.

Other notes

Heightfields Extra Information from GG

Until a more intuitive tool comes along, you will work with the generateChunkFileFromRaw16 script function to create chunk files for Atlas. These are the arguments that it uses:

Argument	Description
srcFile	Filename of the source raw 16 bit heightfield we wish to generate data for. Example: “demo/data/terrains/test_16_4097.raw”.
size	Size of the heightfield, which is assumed to be square. The heightfield will be expected to be ONE MORE THAN THIS. Integer. Size must be a power of 2. Example: 4096.
squareSize	Spacing between sample points, in meters. The default in legacy was 8 meters. This is a floating point number. There are no limitations on this value, although high polygon density may result in reduced frame rates. Example: 2.0. In L3DT terms this is the heightfield resolution.
vertScale	Vertical scale factor. Incoming raw data comes in as a whole number between -32,765 and +32,765, and is scaled by this factor to convert to meters. Factors are commonly of the form 1.0 / 2.0n, n often being between 1 and 8. 1.0 / 256.0 gives you a workable range of about 256 meters in the vertical range. To get an accurate size find the highest point of your terrain and the lowest point. If the lowest is negative add them together. If the lowest is positive then subtract. With your total divide by 65535. Take the first 4 decimal places over and you have your vertical scale. Example: 0.02098
destFile	File to which generated data is written. This file should end in .chu for TGEA’s resource manager to properly load it. Example: “demo/data/terrains/large.chu”.
error	Floating point number indicating the acceptable amount of error for each LOD to possess compared to the previous. This directly and dramatically impacts the number of polygons that will be rendered. Example: 2.0.
treeDepth	Integer indicating how deep the quadtree of chunks is to be. Example: 6.

Map Sizes and comparison

Map Size 1 8192 x 8192 pixels (64 Mpxls) 16 x 16 kilometres

16 x 16 kilometers = 256 kilometers = 159.071025 miles Which is the size of the SWG maps!

Map Size 2 32768 x 32768 pixels (1.0 Gpxls) 66 x 66 kilometres

66 x 66 kilometres = 4 356 kilometers = 2 706.69291 miles

Map Size 2 131072 x 131072 pixels (16 Gpxls) 262 x 262 kilometres

262 x 262 kilometres = 68 644 kilometers = 42 653.4041 miles

Back to tutorial index