Generating Complex Procedural Terrains Using the GPU

Andrzej Kokosza, Weronika Sieińska

January 10, 2019

GPU Gems 3

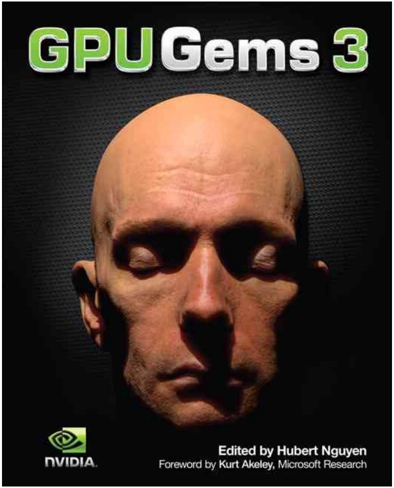

Figure 1: https://books.google.pl/books/about/GPU_Gems_3.html?id=ylNyQgAACAAJ&source=kp_cover&redir_esc=y

Terrain Created Entirely on the GPU

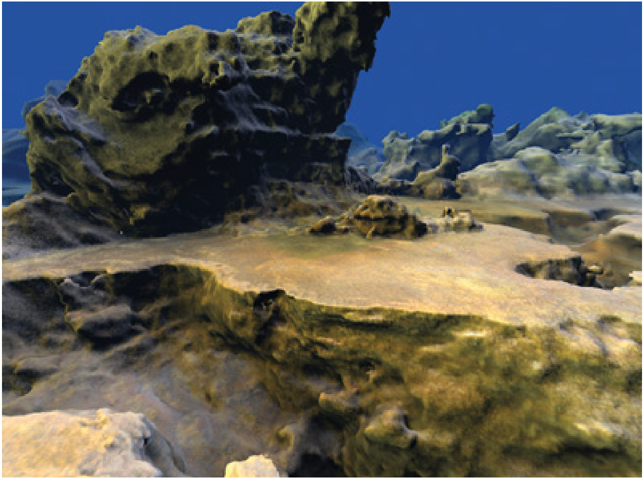

Figure 2: Terrain Created Entirely on the GPU,
https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html

The Density Function

The terrain surface can be completely described by a single function, called the density function.

$$f(x, y, z) = v$$

f – the density function

v – a single floating-point value produced by the density function

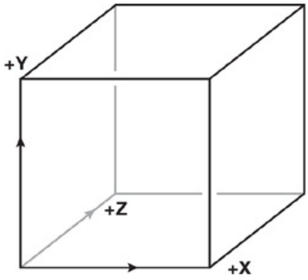

Figure 3: The Coordinate System, `https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html`

The Density Function

We want to construct a polygonal mesh along the surface of the terrain.

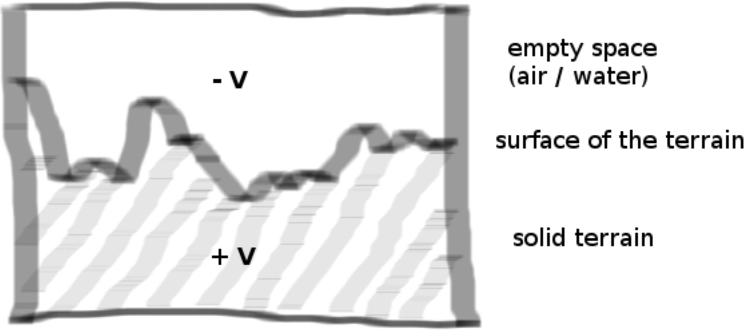

Figure 4: Empty space, surface of the terrain, solid terrain

One cell at a time

- We use the GPU to generate polygons for one cell of terrain at a time.
- Within these cells we will construct polygons (triangles) that represent the terrain surface.
- The marching cubes algorithm allows us to generate the correct polygons within a single cell.

Marching Cubes Algorithm

- The marching cubes algorithm allows us to generate the correct polygons within a single cell.
- INPUT – the density value at eight corners of a cell
- OUTPUT – 0-5 polygons
  - densities at the eight corners of a cell all have the same sign $\rightarrow$ the cell is entirely inside or outside the terrain, so no polygons are output
  - all other cases $\rightarrow$ the cell lies on the boundary between rock and air/water, and 1-5 polygons will be generated

Marching Cubes Algorithm

$v0, v1, ..., v7$ – density values
1 – if the value is positive; 0 – if the value is negative
We then logically concatenate (with a bitwise OR operation) these eight bits to produce a byte.

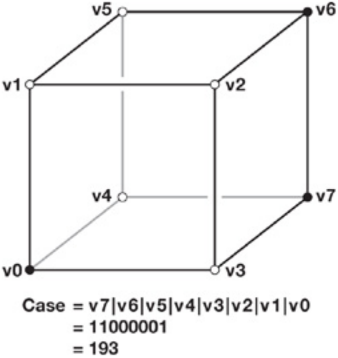

Figure 5: A Single Cell with Density Values,
`https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html`

Marching Cubes Algorithm

- If the byte is 0 or 255, then the cell is entirely inside or outside the terrain and, no polygons will be generated.
- However, if the byte is in the range [1..254], some number of polygons will be generated.
- To determine how many polygons to output various lookup tables are used.
- Each polygon is created by connecting three points (vertices) that lie somewhere on the 12 edges of the cell.

Generating Polygons Within a Cell

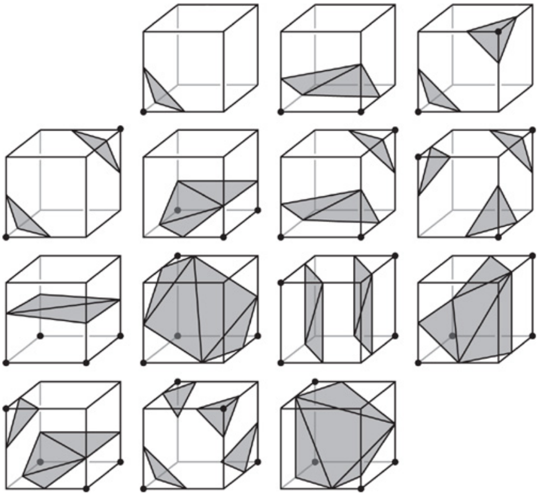

Figure 6: The 14 Fundamental Cases for Marching Cubes, `https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html`

Generating Polygons Within a Cell

The vertex should be placed where the density value is approximately zero. For example, if the density at end v0 of the edge is 0.1 and at end v1 is -0.3, the vertex would be placed 25 percent of the way from v0 to v1.

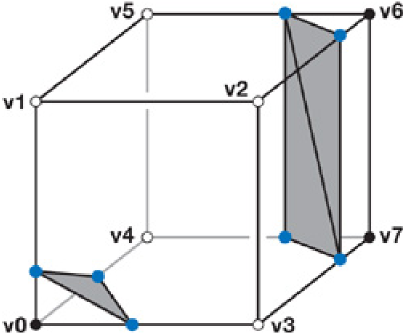

Figure 7: Implicit Surface to Polygon Conversion, https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html

Lookup Tables

1. Lookup Table 1
   - The first table when indexed by the byte number, tells us how many polygons to create for that byte.
   - int byte_to_numpolys[193];
2. Lookup Table 2
   - The second table receives the byte number and provides the information needed to build up to five triangles within the cell.
   - int3 edge_connect_list[193][5];
   - Each of the five triangles is described by just an *int*3 value (three integers); the three values are the edge numbers [0..11] on the cube that must be connected in order to build the triangle.

Lookup Tables

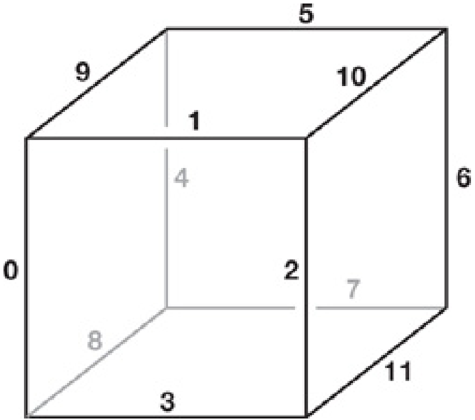

Figure 8: Implicit Surface to Polygon Conversion,
https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html

Lookup Tables

In our example (byte = 193):

int byte_to_numpolys[193]: 3

int3 edge_connect_list[193][0]: 11 5 10

int3 edge_connect_list[193][1]: 11 7 5

int3 edge_connect_list[193][2]: 8 3 0

int3 edge_connect_list[193][3]: -1 -1 -1

int3 edge_connect_list[193][4]: -1 -1 -1

To build the triangles within this cell, a geometry shader would generate and stream out nine vertices.
Note that the last two int3 values are -1; these values will never even be sampled, though, because we know there are only three triangles for this case.

References

GPU Gems 3. Chapter 1. Generating Complex Procedural Terrains Using the GPU.
https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch01.html