

Geometry Clipmaps

Terrain rendering using nested regular grids

Author: Markel Pisano Berrojalbiz

Course: CS562

Instructor: Jon Sanchez

2019-12-12

# Abstract

Rendering virtual terrains from data sets has become a must in many fields, from topography, to film CGI and, of course, videogames. To achieve this goal, sometimes is as simple as dumping all the terrain’s data into the memory card, but most of the times is not that easy.

Many techniques have been developed to deal with this problem, being the most popular those based on adaptable level-of-detail (LOD) based on view. This is the case of Geometry Clipmaps.

Using regular grids and placing them under the view’s camera, this algorithm allows rendering from height maps of any size at a constant computation cost and vertex throughput to the GPU.

# Frank Losasso’s and Hugues Hoppe’s Solution

“Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids” is the name of the original paper this project is based on. The technique was developed by Frank Losasso, from Stanford University, and Hugues Hoppe, from Microsoft Reseach (along with Losasso).

This solution proposes the use of nested regular grids in order to render terrain height maps, maintaining a constant frametime and throughput with the GPU. The only limitation of this solution is the size of the height map dataset, which they mention to use a 40GB height map of the United States. Fortunately they deal with this problem with advanced compression techniques that are not covered in this project.

In their words: *“Geometry clipmaps provide a number of advantages over previous terrain LOD schemes”.* Here are the key ones:

* Optimal rendering throughput. Clipmap vertices reside in memory card, since the same grid can be reused as much as is needed. Also the grid structure is optimal for the cache-reuse.
* Visual continuity. Regions between levels of detail provide spatial continuity just with few instructions in the vertex and fragment shaders.
* Steady rendering. The rendering time is almost constant since is independent of the terrain’s complexity. Only adding LOD layers and incrementing the height map source has negative changes on the frame rate.
* Surface shading. Since the input data is a height map, surface normal can be calculated on the fly.
* Synthesis. Sampled data can be combined with procedurally generated detail.

In summary: *“We present geometry clipmaps, which cache nested rectangular extents of this pyramid to create view-dependent approximations. A unique aspect of the framework is that LOD is independent of the data content. […]Geometry clipmaps unify the LOD management of the terrain geometry and its associated texture signals. The spatially based LOD structure lets low-resolution textures be applied without visual discontinuities at level boundaries.”*

# Author’s implementation

Although this implementation is solely based on Losasso’s and Hoppe’s solution, and even is named the same as the original, it lacks some of the key features. These are the differences:

* No compressing/decompressing technique for input height map data.
* No smoothing between levels.
* No cache friendly grids (if they are, it isn’t on purpose).
* Pre-computed procedural detail instead of dynamic.
* Frustum culling WIP.
* Hardware texture mipmaping instead of view-based.

This version of the algorithm is quite straight forward and can be simplified in 6 steps:

1. Load High Resolution heigh-map. (CPU to GPU)
2. Construct uniform grid meshes. (CPU to GPU)
3. Place grids at different LOD positions and scale. (CPU)
4. Frustrum Culling (CPU, optional).
5. Sample heigh-map using world position as texture coordinates. (Shader, GPU)
6. Modify vertex altitude with the sampled data. (Shader, GPU)

Steps in detail:

1. Load High Resolution heigh-map. (CPU to GPU)

A single channeled texture, preferably of a very high resolution, is loaded into GPU as a byte-per-pixel 2D image. Mipmaps are computed in this step too.



1. Construct uniform grid meshes. (CPU to GPU)

This implementation constructs 4 different types of mesh to complete the mesh of each level. This is due to the generation of a transition region between levels which is called *ring.*For fitting the *ring*, some margin has to be chopped from regular grids. This creates 2 new versions of the regular grid: grid with margin in one side, and grid with margin at the corner (2 side margin).

Red: Central grids, no margin.

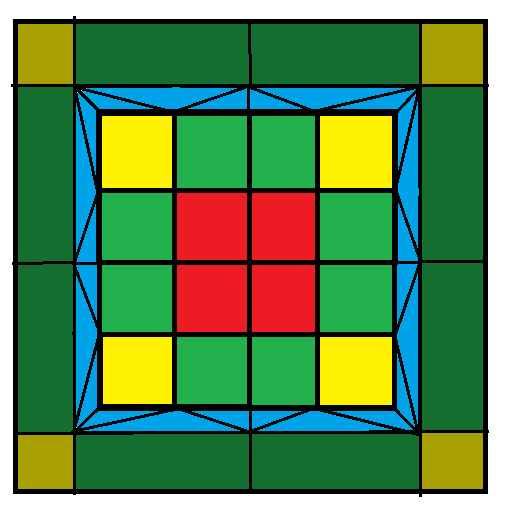
Green: Side grids, one margin.

Yellow: Corner grids, two margins.

Blue: Ring mesh.

Dark Green: Level + 1 side grids.

Dark Yellow: Level + 1 corners.



1. Place grids at different LOD positions and scale. (CPU)
2. Frustrum Culling (CPU, optional).
3. Sample heigh-map using world position as texture coordinates. (Shader, GPU)
4. Modify vertex altitude with the sampled data. (Shader, GPU)

# The Demo

[First check out a few tips to help you quickly format your report. You might be amazed at how easy it is.]

# Other Terrain Rendering Techniques

[First check out a few tips to help you quickly format your report. You might be amazed at how easy it is.]

# Conclusions

[First check out a few tips to help you quickly format your report. You might be amazed at how easy it is.]

# Bibliography

[First check out a few tips to help you quickly format your report. You might be amazed at how easy it is.]

## [Look Great Every Time]

* [Need a heading? On the Home tab, in the Styles gallery, just click the heading style you want.]
* [Notice other styles in that gallery as well, such as for a quote, a numbered list, or a bulleted list like this one.]
* [For best results when selecting text to copy or edit, don’t include space to the right of the characters in your selection.]

### [Add a Quote]

[“Use the Quote style to call out a quotation or other important point from your text.”]

### [Picture Perfect]

[To replace the sample photo with your own, just delete it. Then, on the Insert tab, click Picture.]

[To apply formatting, such as the border and shadow on the sample photo, select your picture. Then, on the Picture Tools Format tab, select a picture style.]