LUNAR GEOMORPHY 3D VISUALIZATION METHOD

Z. Yang, X. Qing, Z. BaoMing, L. JianSheng, L. ChaoZhen

Institute of Surveying and Mapping, ZhengZhou 450052, China – Zhouyang3d@163.com

KEY WORDS: Lunar exploration, Moon Image, Moon DEM, Level of Detail, Visualization

ABSTRACT:

Based on research of large-scale terrain visualization methods, we improve the planar Geometry Clipmaps method by making use of GPU Vertex Processor to projection transform planar terrain into spherical terrain, spherical view culling and spherical viewpoint controlling . We collected and deal with the lunar image and DEM to render the lunar 3D map. The results show that the rendering algorithm' efficiency is independent on datum but there is distort problem in Lunar Pole.

1 INTRODUCTION

Back to Moon, building Lunar base and exploration Lunar resources have been the trend and hot dot of international spaceflight. Lunar exploitation is the first step of Chinese deep space exploration missions. The successful launch of ChangE No.1 satellite indicated that china have the ability to explore the deep space. Obtain lunar remote image and 3D physiognomy data in satellite remote and surveying technology and rendering the 3D map in 3D visualization technology is the one of main tasks of ChangE No.1 satellite. In this paper, based on the research of large range terrain visualization algorithm, we improved the Geometry Clipmaps algorithm and the planar terrain be transformed to the spherical terrain with GPU shaders. We collect the lunar image and DEM and rendered the lunar 3D map by use of the spherical view culling technique and spherical viewpoint control technique to assist human to know well the moon.

2 PREVIOUS WORK

2.1 The terrain render algorithm

A primary difficulty in terrain rendering is displaying realistic terrains to the user at real-time frame rates. Several terrain-rendering techniques have been proposed that use Level of Detail (LOD) to generate a simplified representation of a terrain.

Previous publications and applications can be divided into two parts: Those with static level of detail (S-LOD) and continuous level of detail technique (C-LOD).

(1) S-LOD technique

Here the terrain is divided into tiles each of which is represented by a set of TINs with varying resolutions. Depending on the distance to the viewer for each tile a TIN with appropriate projective triangle size is chosen from the set. If regularly coarsened meshes are used instead of TINs the method is called geo-mipmapping [1].

(2) C-LOD Algorithms

The most elaborate terrain rendering technique known today is the continuous level of detail technique (C-LOD). It improves the sub-optimal approximation quality of the S-LOD algorithms in a sense that the triangulation is altered on a per triangle and not on a per tile basis. This allows much better approximations which adapt optimally both to the viewing distance and to surface roughness.

Several main C-LOD algorithms include Lindstrom [2], Duchaineau [3], Roettger [4], and Losasso[5].

The geometry clipmap is a recently proposed approach that utilizing the potential of modern graphics hardware. The Algorithm caches the terrain in a set of nested regular grids centered about the viewer (fig 1). The grids are stored as vertex buffers in fast video memory, and are incrementally refilled as the viewpoint moves. This simple framework provides visual continuity, uniform frame rate, complexity throttling, and graceful degradation[6].

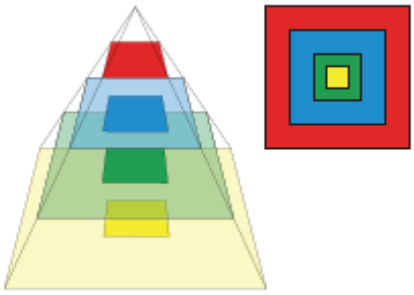

Figure 1: The clipmap contains a fixed-size segment of each mipmap level around an arbitrary focus point. [6]

Those algorithms mentioned in previous section deal with planar terrain. Clasen describe a terrain rendering algorithm for spherical terrains based on clipmaps[7]. The algorithm replaces the underlying geometry with one that maps better to the sphere. No matter how far away the viewer is relative to the planet, he cannot see more of it than one hemisphere. So the algorithm uses concentric rings instead of rectangles. The resulting spherical Geometry Clipmap is displayed in fig. 2.

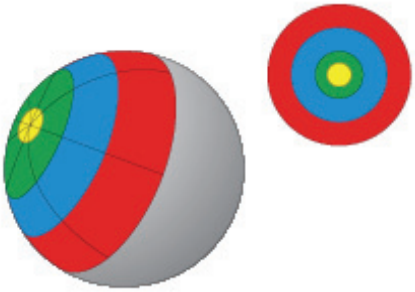

Figure 2: We use circular instead of rectangular rings to cover the hemisphere. [7]

Spherical clipmaps avoid to the terrain distortion in high latitude region and the different levels of detail can be blended smoothly even when they are more than one level apart. But the transformation of the world space (x, y, z) that provides an absolute orientation of the spherical terrain to the view space (x', y', z') that locates the viewer at the lunar pole must be implemented in real time and the concentric rings are transformed into plane irregularly (Fig 3). So reading and updating data is complex and the algorithm is implemented difficultly and inefficiently.

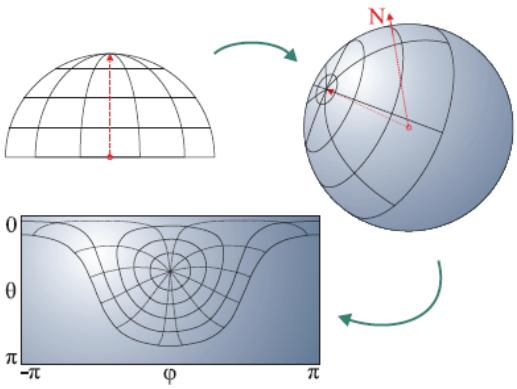

Figure 3: Points on the view hemisphere are transformed into world space to sample the rectangular height map. [7]

2.2 Earth visualization system

With being short of data, terrain rendering focused on earth mainly and the research of visualization in lunar terrain and space environment is absent. At present, the system of 3D digital earth applied successfully include: Google Earth, World Wind, ArcGlobe, and so on. Those systems focus on earth mainly and the digital moon visualization system aren't enough mature and perfect. Google bring forth the Google-Moon in internet[8]. But There are only "Clementine" and "Appolo" remote image and not include DEM data, so there have no the function of 3D visualization. Figure 4 shows the Google-Moon. The NASA brings forth the lunar visualization system- WorldWind-Moon. But the resolution of image and DEM is low and is not meet to the request of application.

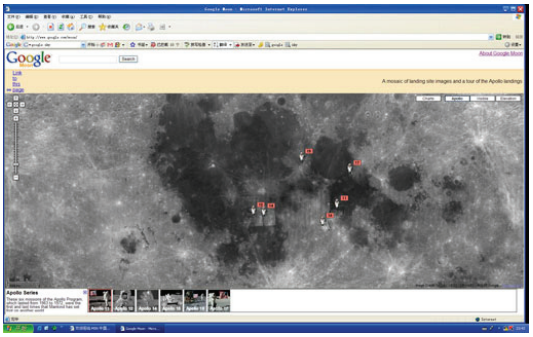

Figure 4: Google-Moon

3 ALGORITHM OVERVIEW AND IMPLEMENTATION

3.1 Geometry Clipmaps

A geometry clipmap renders a set of nested regular grids centred around the viewpoint, with small grids of high detail and large grids of low detail(Fig5). Each grid contains n×n values and is called a clipmap level. The levels are numbered starting from $l = 0$ for the coarsest level. The distance between values at level l is the grid spacing, denoted gl. The vertices in a clipmap level are stored in a vertex buffer on the graphics card. The rectangular rings are divided into 12 parts, for more efficient rendering and view range culling. As the viewpoint moves, the clipmap data is updated so the grids remain centred around the viewer.

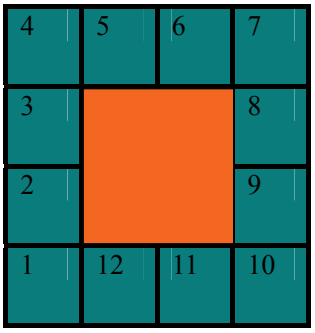

Figure 5: Geometry Clipmaps nested regular grids

The vertices are stored as a toroidal array to enable incremental updates, where only vertices from newly visible areas are added, replacing areas that are no longer visible. Figure 6 shows how toroidal arrays make incremental updates possible. The heightmap and viewer position are shown, as well as the actual clipmap level data. Suppose the viewer is positioned as in Figure 6(a). If we move to the southeast, as shown in Figure 6(b), only the newly visible areas along the bottom and right edges of the heightmap need to be put in the array, and they are put in the top and left edges of the clipmap, respectively, overwriting the data that is no longer needed.

With the use of ring array and mod operation, after the transformation, the vertex's position is no change in array. For example, if n =129, the vertex's number a is 0 → 128. When viewpoint is move and the

number is changed to 1→ 129, the mod(129,129)=0.

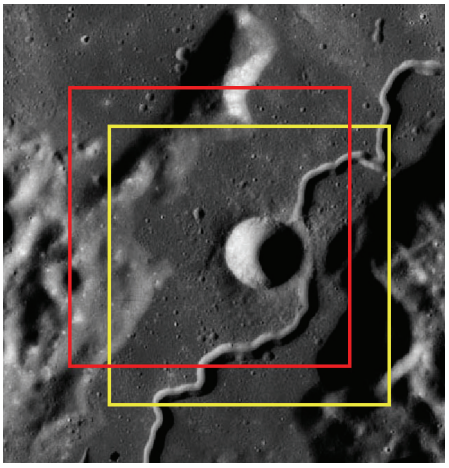

Figure 6 (a): Before a change in viewpoint

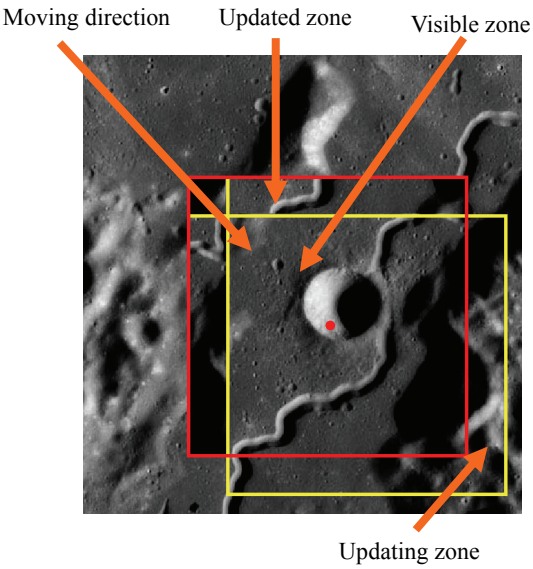

Figure 6 (b): After a change in viewpoint to the southeast

Figure 6: An example of the data in the heightmap (top) and toroidal array (bottom).before and after a change in viewpoint. The position of the viewer in the heightmap is shown by the red dot.

3.2 Transformation of planar terrain to spherical terrain

The test data is Lunar DEM and image in WGS84 coordinate system. If the coordinate of a grid point is (B, L, H), where B is longitude, L is latitude, and H is elevation. We must transform the WGS84 coordinate system to OpenGL world space coordinate system for spherical terrain rendering.

According to Eq.(3), we can transform the WGS84 coordinate system to OpenGL world space coordinate system.

$$\begin{cases} X = (N+H)\cos B\cos L \\ Y = \left[N(1-e^2)+H\right]\sin B \\ Z = (N+H)\cos B\sin L \end{cases} \quad (1)$$

where $\begin{cases} N = \dfrac{a}{\sqrt{1-e^2\sin^2 B}} \\ e^2 = \dfrac{a^2-b^2}{a^2} \end{cases}$

$a = 1738000\text{m}, b = 1737400\text{m}$, a is lunar long radius, and b is short radius. For reducing the CPU' burden and optimizing efficiency, we can implement the equation by GPU.

3.3 Spherical view range culling

As shown in Figure 7, for optimizing efficiency, we use view range culling and back face culling algorithm to eliminate invalid data.

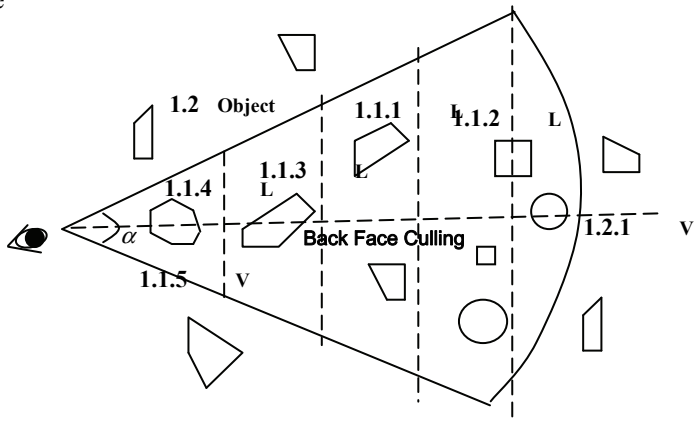

Figure 7: Data culling based on viewpoint and visible face

As shown in Figure 8, the spherical terrain can be divided into two parts. One part is face to viewpoint and one part is back to viewpoint. The $a'b'$ in back to viewpoint part is in view cone, but it is invisible to viewer. So we must eliminate it with spherical view range culling algorithm:

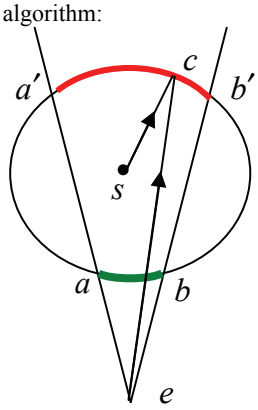

Figure 8: spherical view range culling

$$\begin{cases} \overrightarrow{sc} \cdot \overrightarrow{ec} > 0 \ , \ invisible \\ \overrightarrow{sc} \cdot \overrightarrow{ec} \le 0 \ , \ visible \end{cases} \quad (2)$$

Where, s is the center point of earth, e is viewpoint, and c is a point on spherical surface in view range. If the angle α from vector sc to vector ec $\le 90^0$, point s is invisible. Whereas point s is visible.

As shown in Figure 9, We implement Geometry Clipmaps algorithm with spherical culling, the efficiency be shown by blue dashed. The red dashed show the efficiency of Geometry Clipmaps algorithm without spherical culling. The x-coordinate is the deferent viewpoint and the y-coordinate is render fps. Compared the result, we can find the Geometry Clipmaps algorithm with spherical culling is more efficient than Geometry Clipmaps algorithm without spherical culling. The average frame of Geometry Clipmaps algorithm with spherical culling is 23 fps and the average frame of Geometry Clipmaps algorithm without spherical culling is 9 fps.

Analyzing the test result, the render efficiency of Geometry Clipmaps is steady. Using Geometry Clipmaps algorithm with spherical culling, because we must read different data file in hard disk, when viewpoint ramble to boundary of data block, the rendering efficiency is low. For example, a point in Figure 9. Because the efficiency of Geometry Clipmaps algorithm without spherical culling is low, so the data file's searching and reading don't influence the rendering efficiency.

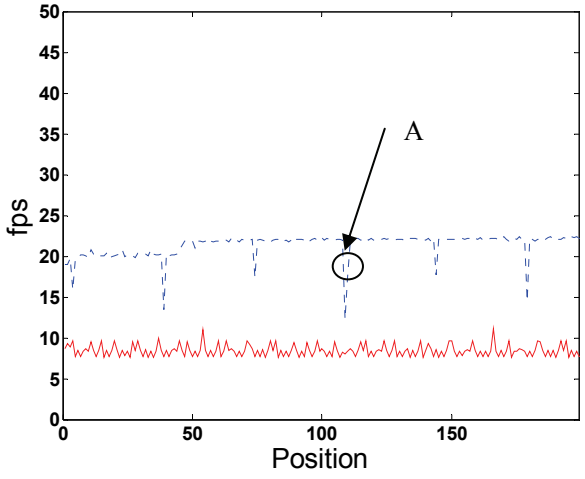

Figure 9: Comparing the rendering frame of view range culling

3.4 Viewpoint control

Rambling in spherical surface, the controlling of view point is more complex than ramble in plane. As shown in Figure 10, if we rotate the line of sight n in viewpoint P, we can rotate the line of sight n about vector OP. Defining the great circle from point P_1 to P_2 is AP_1P_2A'. Translating P_1 to P_2, we can rotate vector OP_1 about great circle's normal OY. As shown in Figure 10, A θ rotation of vector V about vector n should produce the vector V'. The Equation is [12]

$$\vec{V}' = \vec{V}.\cos\theta + \vec{V}.\vec{n}.(1-\cos\theta)\vec{n} + (\vec{V}\times\vec{n}).\sin\theta \quad (3)$$

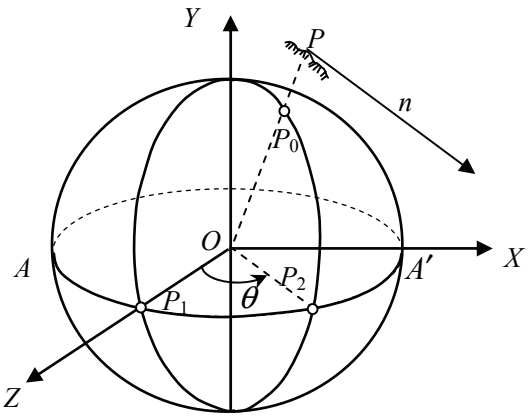

Figure 10: Transformation of spherical view point

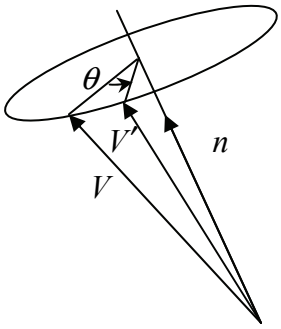

Figure 11: Rotating vector about pointed axis

4 RESULTS AND DISCUSSION

4.1 Test Data and Results

The test data was from USGS's web. The global lunar digital elevation models (DEMs) is at a resolution of 16 pixels/degree (e.g. about 1.895 km resolution). As shown in Figure 12, the size of DEM grid is 5760×2880, created from a triangle irregular network (TIN) of the original points-Unified Lunar Control Network (ULCN2005). See Tables 1 for statistics on this and the other networks.

The global lunar image data is Clementine UVVIS(5 bands, 100m/pixel). The size of data is 163840×81920. In addition, there are high resolution Appolo15 image and DEM in the zone of Appolo15 land in moon. The resolution of Appolo15 image is 1.5m/pixel, the size of data is 3319×3226 and grid cell size is 50m.

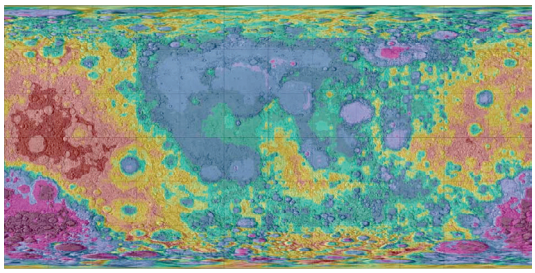

Figure 12: Lunar DEM

The spherical Geometry Clipmaps request the DEM size equal to image half size in same LOD layer. For example, if image size is 1024×1024, then the DEM size must be 512×512. So we must resample the low resolution DEM data with bi-linear interpolation method.

Finally, we built the pyramid of DEM and image. The amounts of pyramid layer are nine. The test results are shown in Figure 13 to Figure 17.

Figure 13: Lunar front face 3D map

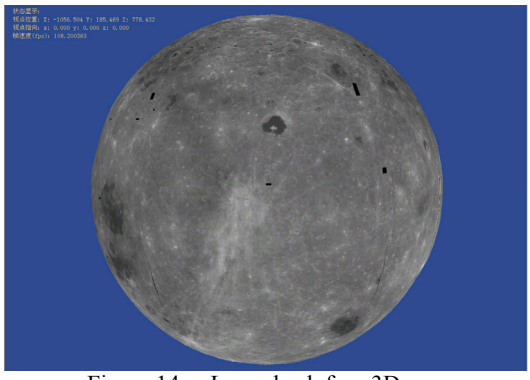

Figure 14: Lunar back face 3D map

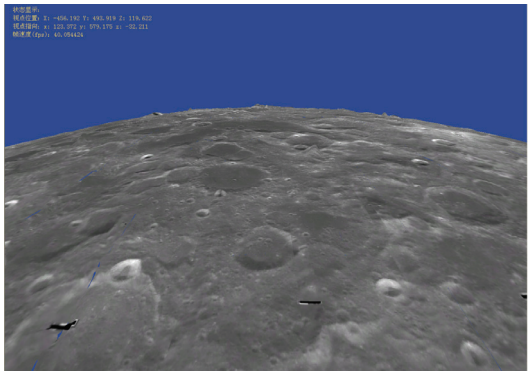

Figure 15: Lunar local zone 3D map

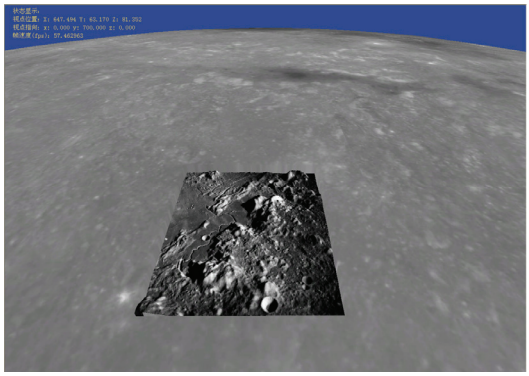

Figure 16: Different resolution mosaic 3D map

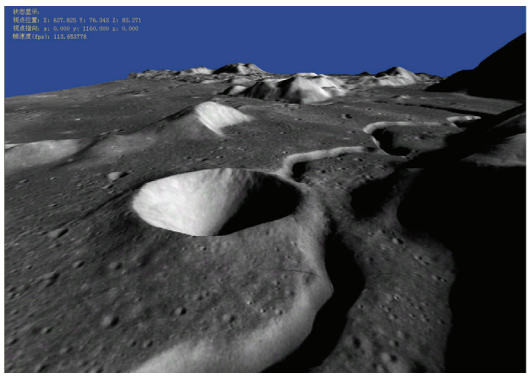

Figure 17: The high resolution 3D map in Applo15 zone

4.2 Conclusion and Future Work

We collected Lunar DEM and remote image, eliminated data's gross error, resampled the data, built the data pyramid, and rendered the global lunar 3D map real time. The frame of rendering is independent on size of data. The future works include:

(1) The solution of distortion in lunar pole

Using the Eq.(1) performing in GPU, We can transform the planar terrain to spherical terrain. But the problem of projection distortion in lunar pole will worsen the rendering effect. The projection result be shown in Figure 18. Because there are best zone to human explore moon and build lunar base, so we must optimize the LOD and projection algorithm to improve the rendering result.

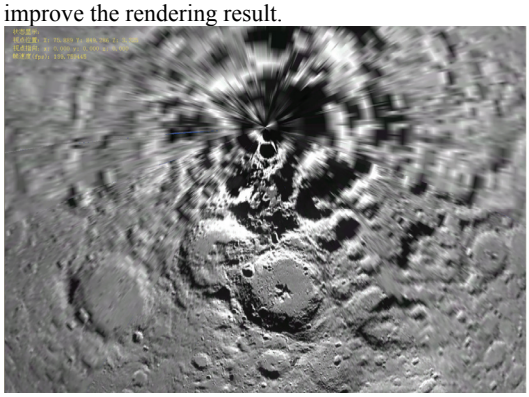

Figure 18: projection distortion in lunar pole

(2) Higher precision lunar terrain rendering

With the restriction of data gained means, the lunar

terrain and image's resolution is low. So the rendering result is not satisfied. New means of data gained must be developed.

(3) The parallel visualization of multi-planet

The rendering algorithm presented in this paper focused on single planet. Building 3D virtual space environment, we must realize the parallel visualization and seamless rambling on earth, moon, mars, and so on. So we must study on space-time reference frame, coordinate system transformation, data structure, view range culling, data storage and searches.

ACKNOWLEDGEMENTS

We would like to thank Kan Ning, Zhang Yong, Cheng Jinwei and Yao ZhiQiang for the work of dealing with data and testing algorithm. We also appreciate the publication of the Lunar DEM and image data by NSGS (http://pubs.usgs.gov/of/2006/1367/dems/).

REFERENCES

[1] DE BOER, W. H. Fast Terrain Rendering Using Geometrical Mipmapping. E-mersion Project (2000).

[2] LINDSTROM, P., KOLLER, D., RIBARSKY, W., HODGES, L. F., FAUST, N., AND TURNER, G. Real-Time, Continuous Level of Detail Rendering of Height Fields. In Proc. SIGGRAPH '96 (1996), ACM, pp. 109–118.

[3] DUCHAINEAU, M., WOLINSKY, M., SIGETI, D. E., MILLER, M. C.,ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. ROAMing Terrain: Real-Time Optimally Adapting Meshes. In Proc. Visualization '97 (1997), IEEE, pp. 81–88.

[4] ROETTGER, S., HEIDRICH, W., SLUSALLEK, P., AND SEIDEL, H.-P. Real-Time Generation of Continuous Levels of Detail for Height Fields. In Proc. WSCG '98 (1998), EG/IFIP, pp. 315–322.

[5] LOSASSO, F., AND HOPPE, H. Geometry clipmaps: terrain rendering using nested regular grids. ACM Transactions on Graphics (2004), 769–776.

[6] HOPPE, H. Terrain Rendering Using GPU-Based Geometry Clipmaps, ACM Transactions on Graphics (2004).

[7] MALTE, C. HANS-CHRISTIAN, H. Terrain Rendering using Spherical Clipmaps, Eurographics/ IEEE-VGTC Symposium on Visualization (2006)

[8] http://www.google.com/moon/