Level of Detail Generation And Texture Mapping

A Literature Review

Benjamin Meier
MRXBEN001

University of Cape Town
Department of Computer Science

May 15, 2014

Abstract

This review describes methods of using level of detail (LOD) algorithms and texturing to support real time rendering of very large models produced by the Zamani project at the University of Cape Town. The models produced by the project contain millions of triangles and are often too dense and complex to render in real-time on current consumer hardware and so require a method of reducing the amount of geometry being rendered. Techniques covered include discrete level of detail, continuous level of detail, and view-dependant level of detail techniques such as as progressive meshing and progressive buffers. Normal mapping and displacement mapping are considered as methods to recover detail lost during simplification.

1 Introduction

The Zamani project run by the Geomatics Department at the University of Cape Town aims to capture accurate three dimensional architectural models of African heritage sites for storage in the African Cultural Heritage and Landscape Database.

The project uses phase based laser scanners to record range data from 300 - 800 positions around a site. Buildings are typically scanned with 1 - 2 cm point spacing while terrain is captured with 10 - 50 cm point spacing [RHB+12]. The range information is used to determine a dense point cloud that represents the surface, building, or terrain being scanned. The point cloud is cleaned by hand of any unwanted objects or points before undergoing surface reconstruction to determine a continuous surface of triangles which can be very detailed and contain billions of triangles. The researchers use mesh processing software such as MeshLab and ArcGIS to navigate and interact with the resulting models but the models are often too large to render at interactive frame rates making manipulation difficult. Dense point clouds are often decimated to reduce the amount of resulting geometry and thus increase rendering speed however this has harmful effects on details such as crevices and surface roughness which are particularly important to researchers.

This literature review serves as a survey of current techniques for real time rendering of very large 3D meshes and their appropriateness in the Zamani project space. Specifically we will look at level of detail schemes and the use of texturing to recover surface detail.

2 Previous Work

Over the past 30 years, many techniques of optimising rendering of very large models have been put forward. Primarily this involves simplification of the model in order to reduce the number of triangles being rendered. This simplification occurs as a preprocessing step and results in a copy of the model containing less geometry, unfortunately small details are important to the Zamani project and a method is required that conserves very small details in the mesh.

2.1 Level of Detail Systems

Level of detail (LOD) algorithms are commonly used in computer graphics to reduce the complexity of 2D or 3D objects as they move away from a virtual camera that is using a perspective projection. Because the object becomes smaller on the screen the effect of the reduction in complexity is very small. LOD techniques have been used

to reduce the effect of aliasing on 2D textures since 1983 [Wil83] and have become common in video games for managing detail in large 3D terrains.

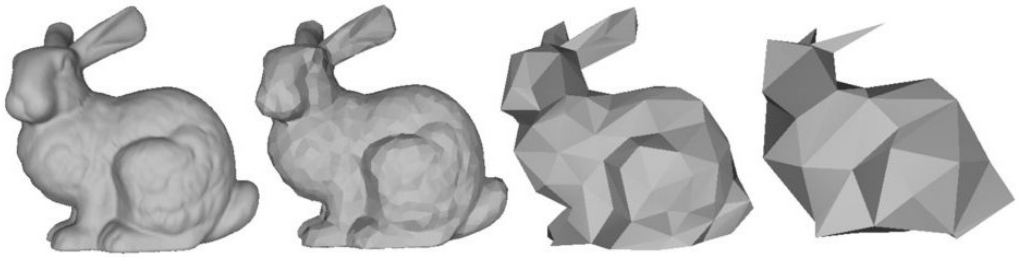

Figure 1: A demonstration of levels of detail. The bunny on the left contains 69,451 triangles while the bunny on the right contains only 76.

Source: Stanford 3D Scanning Repository

2.1.1 Discrete Level of Detail

Two main LOD techniques exist for managing level of detail: Discrete LOD (DLOD) and Continuous LOD (CLOD) [Ali10][RLB10]. Discrete level of detail is the technique most commonly used in 3D graphics and is a form of 3D mipmapping [Wil83]. In this approach, a small number of discrete levels of detail for a model are stored at fixed resolutions with fixed numbers of triangles. These are generated as a preprocessing step. These specific LOD models are swapped into the scene at certain distances from the camera in order to maintain the illusion of having the same complexity. Advantages of DLOD are that the models can be rendered very quickly and can be further preprocessed to refine textures and rebind animations. The major disadvantage of DLOD is the 'popping' effect seen when models of a higher resolution are suddenly swapped into a scene with no blending from their low resolution counterpart. This is often seen when the camera is moving very quickly towards the object. A naive solution to this is to simply generate more levels of detail, however this takes up more space on disk and requires more memory transfer to the graphics hardware. Lower levels of detail can also exhibit bad silhouettes when convex surfaces or edges are simplified [Ali10] as can be seen around the last level of detail in Figure 1.

2.1.2 Continuous Level of Detail

Continuous LOD is an alternative LOD process which allows continuous variability in the resolution of the model at run time with a high degree of accuracy. CLOD can reduce the number of triangles used in a model in a continuous manner when less detail is required. The continuous nature of CLOD makes it immune to the 'popping' effect seen in DLOD [RLB10] and models use no more geometry than is required. The net result is higher fidelity due to increased granularity [Ali10]. CLOD requires specific data structures from which the desired amount of detail can be extracted at run time [RLB10]. Progressive Meshes [Hop96] are a CLOD datastructure which stores a coarse simplification of the mesh alongside an ordered list of geomorph operations that can be applied to recover the original mesh in high detail. This requires more memory than then the original mesh since each operation needs to be explicit regarding how and where an edge is rebuilt.

2.1.3 View-dependant Level of Detail

View-dependant Level of Detail (VLOD) is an extension of CLOD that allows continuous variability in the geometry of arbitrary regions of a mesh. This is view-dependant as geometry far away from the camera can be progressively simplified while geometry closest to the camera is made more dense. Single objects can span multiple levels of detail and VLOD allows the silhouette of an object to be shown at higher detail than interior regions. VLOD enabled drastic simplification of very large models while still allowing any area to be shown at its highest detail level. The main disadvantage of VLOD is the increased memory usage and preprocessing time. View Dependant Progressive Meshing [Pri00] and Progressive Buffers [SM05] are examples of VLOD.

The disadvantage of techniques based on progressive meshing is that they require additional storage space, a large preprocessing time, and a large amount of memory and video memory at runtime. Chris Prince developed [Pri00] an extension of the progressive mesh data structure in 2000 that allowed geometry to be recovered anywhere in very large models. Using hierarchical simplification and careful optimisations he demonstrated a technique for rendering very large progressive meshes without keeping the entire model in memory. Although the largest model he used contained only

11 million triangles, his approach should still be viable for larger models on modern hardware.

In 2005, Pedro Sander and Jason Mitchell presented a further improvement of progressive meshes using progressive buffers [SM05]. This data structure uses a sequence of discrete LOD meshes for different clusters of polygons in a large mesh. Continuous transformation between discrete levels of detail anywhere in the mesh is achieved using geomorphing and allows dynamic texturing of models with support for mipmapped textures. Support for texture mapping is important in the context of the Zamani project but Sander and Mitchell note that texture mapping degrades rendering performance considerably as it raises the number of drawing called sent to the graphics hardware. Sander and Mitchell still achieved good results and could render an architectural scene of 14 million triangles with a moving camera at a consistent 60 frames per second (fps).

An alternative improvement on progressive meshing was developed in 2003 by Paolo Cignoni et. al and utilises transformations of preprocessed triangle patches and tetrahedra [CGG+04]. A hierarchical tetrahedra data structure allows specific refinement of details and is not limited to a specific topological genus or connectedness. The technique is GPU bound and produced effective results with a large 373 million polygon model on consumer hardware of the time. The disadvantages of this technique included very long preprocessing times (20 hours for the aforementioned model) and higher memory consumption (roughly a factor of 3 over the original model). This technique did not support texture mapping.

2.1.4 Hierarchical Level of Detail

Discrete LOD is also suboptimal when applied to large models that are close to the camera and span a large area in the 3D scene [Lak04]. The short distance to the camera means the model needs to be loaded and rendered at full resolution, but this is wasted when much of the new dense geometry is far away from the viewer or outside of the view frustum. A method was devised to mitigate this by splitting a mesh up as an octree data structure whereby each node of the octree is a simplification of its 8 child nodes [Lak04]. Each submesh node can be highly optimised as a triangle strip

and can be texture mapped. Hierarchical LOD (HLOD) also reduces the required memory bandwidth to the graphics card at run time since only small sections of the mesh need to be reloaded when nodes are collapsed or split as the camera moves around the mesh. Ali Lakhia developed an efficient and predictive memory manager that dynamically loads and unloads submeshes using a priority queue mechanism. Parallelism was used to load mesh data from secondary storage without blocking the rendering thread [Lak04].

2.2 Using Texture to Recover Lost Surface Details

Wrinkles and surface features can be represented on a 3D model in various ways without creating new geometry.

2.2.1 Bump Maps

Bump maps were first developed by James Blin in 1978 paper: Simulation of Wrinkled Surfaces [Bli78]. Bump maps modify the surface normals of an object by providing a height map texture for an object. This height map models vertical displacement of points on the surface. Lighting calculations calculate a new normal for a point by calculating the slope on the height map at that point and using that for reflections. Typically bump map textures are greyscale and restricted to only 256 different values however 16 bit and 32 bit bump mapping is possible as well. The disadvantage of bump mapping is that normals need to be calculated at runtime from slope information given by the texture. This can be relatively complex. Bump mapping only appears to modify the geometry and thus runs into problems on the silhouette of an object where the surface appears perfectly flat when viewed from an acute angle.

2.2.2 Normal Maps

In 1998 the idea of mapping surface normals from a high resolution mesh to a lower resolution mesh was presented as a method of preserving surface detail in simplified models [COM98]. In this paper, normals were stored directly in textures and could be easily applied and interpolated across a surface. The normals are encoded as their X, Y, and Z components with 8 bits per channel and can be combined with the existing

surface normal at a point using an inexpensive dot-product operation. Normal maps are visually similar to bump maps when applied to models but are harder for a human artist to create by hand since 3 dimensions are represented at each pixel in the texture rather than the one dimension that bump mapping uses. Normal maps encounter the same silhouette problems that bump maps do since the normal is simply adjusted without changing the underlying surface. Very fast techniques have been developed to map high resolution normals on low resolution models using GPUs [GGC06]. This was only tested on models with up to 60 thousand triangles and not millions as the Zamani project uses. Normal mapping could be very effective for conserving surface information when simplifying high resolution laser scanned models.

2.2.3 Displacement Maps

Displacement mapping was first introduced by Robert Cook in his 1984 paper: Shade Trees [Coo84]. As well as developing many of the concepts used in modern lighting simulations, Cook explained the concept of a displacement map that could be used to displace geometry along the surface normal. Modern shaders implement this by subdividing triangles a number of times and displacing the resulting vertices according to the values found in the height map texture. This also effectively modifies the surface normal at that point on the original surface.

Displacement mapping provides a solution to the silhouette issues encountered by bump maps and normal maps because actual geometry is displaced [Coo84]. Displacement mapping is relatively performance intensive on the GPU as more triangles need to be drawn but is more efficient than modeling the actual geometry in most cases. Many different approaches have been developed for displacement mapping that trade rendering speed for accuracy and vertex displacement using vertex shaders has been determined to be the most accurate at close camera distances, while fragment shader solutions are faster to render [SKU08].

Figure 2: The visual difference between the original surface (left), with a normal map applied (center), and with a displacement map applied (right).

Source: www.chromesphere.com

3 Conclusion

To conclude, we see that a large amount of research has been done into the rendering of models with very large amounts of triangles over the last 30 years. As consumer hardware performance increases, more powerful preprocessing algorithms become viable. We see that a careful balance must be maintained between the amount of memory required during preprocessing and rendering; the detail and realism that results; and the resulting rendering rate. Some techniques such as progressive buffers and hierarchical tetrahedra rendering are certainly more algorithmically complex than some alternatives although they seem to have the best performance based on experimental results of the time.

Normal mapping is an appropriate technique for baking high resolution surface details onto a simplified version of the model and can be easily integrated into existing Hierarchical Level of Detail structures implementations. Care will need to be taken in choosing a simplification and level of detail technique to use when developing a renderer for Zamani project models since many of these solutions are very memory or CPU intensive.

List of Figures

1 A demonstration of levels of detail. The bunny on the left contains 69,451 triangles while the bunny on the right contains only 76. 3
2 The visual difference between the original surface (left), with a normal map applied (center), and with a displacement map applied (right). . 8

References

[Ali10] Daniel Aliaga. Level of detail: A brief overview. http://www.cs.purdue.edu/homes/aliaga/cs535-10/lec-lod.pdf, 2010.

[Bli78] James F. Blinn. Simulation of wrinkled surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '78, pages 286–292, New York, NY, USA, 1978. ACM.

[CGG+04] Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. Adaptive tetrapuzzles: Efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Trans. Graph.*, 23(3):796–803, August 2004.

[COM98] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *IN PROC. SIGGRAPH'98*, pages 115–122, 1998.

[Coo84] Robert Cook. Shade trees. *Computer Graphics*, 18(3):223–231, 1984.

[GGC06] Jesús Gumbau, Carlos González, and Miguel Chover. Fast gpu-based normal map generation for simplified models. In *Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2006*, 2006.

[Hop96] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM.

[Lak04] Ali Lakhia. Efficient interactive rendering of detailed models with hierarchical levels of detail. In *3DPVT*, pages 275–282. IEEE Computer Society, 2004.

[Pri00] Chris Prince. Progressive meshes for large models of arbitrary topology. Master's thesis, University of Washington, Seattle, 2000.

[RHB+12] Heinz Rüther, Christof Held, Roshan Bhurtha, Ralph Schroeder, and Stephen Wessels. From point cloud to textured model, the zamani laser scanning pipeline in heritage documentation. *South African Journal of Geomatics*, 1(1):44–59, 2012.

[RLB10] José Ribelles, Angeles López, and Oscar Belmonte. An improved discrete level of detail model through an incremental representation. In John P. Collomosse and Ian J. Grimstead, editors, *TPCG*, pages 59–66. Eurographics Association, 2010.

[SKU08] László Szirmay-Kalos and Tamás Umenhoffer. Displacement mapping on the gpu - state of the art. *Comput. Graph. Forum*, 27(6):1567–1592, 2008.

[SM05] Pedro V. Sander and Jason L. Mitchell. Progressive buffers: View-dependent geometry and texture for lod rendering. In Mathieu Desbrun and Helmut Pottmann, editors, *Symposium on Geometry Processing*, volume 255 of *ACM International Conference Proceeding Series*, pages 129–138. Eurographics Association, 2005.

[Wil83] Lance Williams. Pyramidal parametrics. *SIGGRAPH Comput. Graph.*, 17(3):1–11, July 1983.