Fourth LACCEI International Latin American and Caribbean Conference for Engineering and Technology (LACCET'2006)
"Breaking Frontiers and Barriers in Engineering: Education, Research and Practice"
21-23 June 2006, Mayagüez, Puerto Rico.

Issues in Terrain Visualization for Environmental Monitoring Applications

Ricardo Veguilla, BSCpE
Graduate Student, ECE Dept., University of Puerto Rico, Mayagüez Campus,
veguilla@ece.uprm.edu

Nayda G. Santiago, Ph.D., PE
Assistant Professor, ECE Dept., University of Puerto Rico, Mayagüez Campus,
nayda.santiago@ece.uprm.edu

Domingo A. Rodriguez, Ph.D.
Professor, ECE Dept., University of Puerto Rico, Mayagüez Campus,
domingo@ece.uprm.edu

Abstract

For a more effective study of the interaction between the different natural and human-caused enviromental phenomena that affect the Jobos Bay Natural Reserve we envision the integration of remotely gathered images and elevation data from satellite and aerial sensors, with GIS and real-time data from wireless sensors into a unified interactive 3D visual representation of the reserves assets.

Performing accurate visualization of terrain interactively usually require expensive high-end computer systems since the size of the raw data sets involved exceed the limited storage and/or processing capacity available in common computers. By employing rendering level-of-detail techniques (LOD), it is possible to reduce the data to a manageable size while retaining an acceptable level detail where required, eliminating the need for high-performance computing system for visualization.

In this work, we will review the traditional and state-of-the-art techniques for level-of-detail rendering of terrain data. Particular interested will be given to a) recent work on how to better exploit the processing capabilities of consumer-lever graphic processor units, b) support in LOD techniques for out-of-core terrain data management for handling massive terrain data sets. Our purpose is to identify the the most convinient LOD approaches for a future implementation as part of an ongoing development of a interactive terrain visulization system for environmental monitoring applications

Keywords

Terrain visualization, level of detail, out-of-core algorithms, GPU.

1. Introduction

There has been an historical struggle between complexity and performance in the field Computer Graphics. Despite the enormous advances in graphics hardware rendering capabilities, the complexity (usually measured in the number of polygons used to represent a particular object) seems to grow faster than ability of the hardware to render them. This problem is critical interactive terrain visualization and is generally caused by the sheer size of the geometry to be rendered, the suboptimal organization of the data, or the limited rendering capabilities of the available graphic hardware. The discipline of level-of-detail (LOD) attempts to balance complexity and performance by regulating the amount of detail used to represents objects based on an geometric and/or visual fidelity error metric in order to maintain a faithful representation of the terrain for interactive rendering.

The basic concept behind LOD (Clark, 1976) is to use less detail for small, distant portions of the scene to be rendered and is based on the observation that it is inherently inefficient to use many polygons to render object that will only contribute to a few pixels of the rendered scene. For clarity, in this paper we will refer to a level of detail management algorithm as a LOD algorithms, and we will refer to the polygonal mesh of an given detail resolution produced by the LOD algorithm as a LOD.

In Section 2 we will present the basic characteristics of the different LOD techniques used for terrain visualization. In Section 3 we will discuss common problems and considerations that affect the design of LOD techniques. In Section 4 we will present a survey of the LOD techniques for terrain visualization. In section 5 we describe our ongoing efforts to develop a terrain visualization system for enviromental monitoring called VTE. Finally we present our conclusions in Section 6.

2. Characteristics of LOD algorithms

An excellent in-depth overview of the general problem of LOD can be found in (Luebke, et al., 2002). In this section we will summarize the most relevant aspects and considerations related to LOD for terrain visualization.

In general terms, a LOD algorithm is responsible for performing geometric simplification operation to eliminate redundant information where appropriate while satisfying both performance and visual accuracy constrains. The basic components of a LOD algorithms are the *initial mesh*, which consist of representation of the terrain at either the minimum resolution level or the maximum resolution level, and a set of *updates operations*, that, when applied to the initial mesh, produce a corresponding mesh of different resolution. Historically, the mesh with the minimum number of polygons required to approximate the full resolution terrain has been called the *base mesh*. We will refer to the full resolution representation of a terrain as the *full mesh*. Even though the a LOD algorithm can begin with a base mesh (simplest representation) and progressively add detail, the overall process is still one of geometric simplification with respect to the original full mesh, since the resulting mesh is a simplified version of the original.

LOD algorithms can be classified based on the following aspects:

2.1 LOD granularity: discrete or continuous

A LOD algorithm can be classified as either *discrete* or *continuous* depending on the granularity between successive LODs used by the algorithm. In discrete LOD algorithms, multiple individual models of different resolutions are generated from the terrain data during an off-line processing operation, and the appropriate model is selected at runtime. In continuous LOD algorithms, a continuous spectrum of detail for the terrain is encoded in a data structure from which a model for a desired level of detail can be extracted at runtime. Since the bulk of the geometric simplification is performed off-line, discrete LOD algorithms are simpler and their runtime overhead is usually limited to evaluating the selection criteria and handling transitions between LODs. Continuous LODs algorithms are more complex, incur in considerable more runtime overhead but theoretically are more efficient since providing more granularities between LODs should allow for more efficient resource utilization because the algorithm should only use as many polygons as required to achieve the desired LOD.

2.2 LOD distribution: uniform or view-dependent

Depending on the distribution of geometric complexity across the resulting mesh a LOD algorithms can be classified as performing *uniform* or *view-dependent* geometric simplification. In uniform simplification algorithms, the level of detail of a resulting mesh is uniform across the geometry, whereas in view-dependent simplification algorithms the level of details varies with respect to the view direction and the region of the mesh contained inside the view frustum. Uniform simplification is usually employed in discrete LODs generation which is performed off-line when no view-dependent information is available. View-dependent simplification can provide considerable benefits for terrain visualization since it will maintain detail in mesh regions within the view frustum and facing the view direction, and eliminate detail on regions outside the view frustum and/or facing away from the view direction, while at the same time, allowing transition between different LODs through the continuous region of the visible terrain mesh.

2.3 Processing direction: top-down or bottom up

As mentioned before, the geometric simplification operation performed by a LOD algorithm begins with using either the base mesh (simplest representation) or the full mesh (most complex representation) as the initial mesh. As a result, the use initial mesh selection allows us to classified the algorithms in terms of the direction in which update operation introduce complexity into the resulting mesh. A *top-down* LOD algorithms (also referred to as *refinement* or *subdivision* algorithm) begins with a base mesh and then proceed to progressively add vertices, incrementing complexity, until the desired resolution is achieved. A *bottom-up* LOD algorithms (*simplification* or *decimation*) begins with a full mesh and proceed to progressively remove vertices (decreasing complexity) until the target resolution is reached. Bottom-up algorithms have higher memory and computational cost since they start with the full mesh, but are able to find the minimum number of polygons for a given accuracy level. As a consequence, bottom-up algorithms are generally used during off line discreet uniform simplification operations. Top-down algorithms are ideally suitable for run-time operations since they complement view-dependant simplification algorithms by supporting view culling i.e. discarding polygons that lie outside the view frustum.

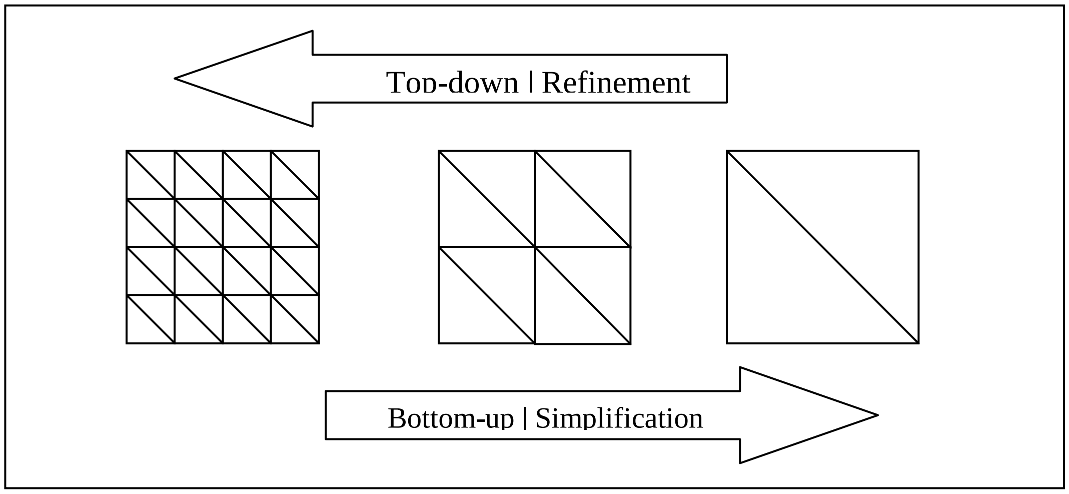

Figure 1: Terrain Refinement/Simplification Process.

2.4 Terrain data structure: regular or irregular

LOD algorithms are also differentiated by the structure employed to represent the terrain. The basic two representations are height fields and triangulated irregular networks (TINs) (Fowler and Little, 1979). Height fields consist of an array of height values at regularly spaced *x* and *y* coordinates. In TINs, height values are irregularly spaced. In general, height fields are considered less efficient than TINs because they store redundant information in regions with constant elevation change, whereas TINs can use the minimum number of elevation samples to approximate a terrain by using few samples to describe large flat areas while using many samples on areas of larger terrain complexity. In terms of management, height fields are easier to work due to their simple spatial organization.

2.5 LOD data structure: quadtrees, bintrees, LOD pyramid

In order to implement view-dependent LOD a hierarchical structure capable of representing different parts of the terrain at different resolutions is required. The most commonly used structures for this purpose are the *quadtree* and the *bintree* (Samet, 1984). In a quadtree structure, a rectangular region recursively subdivided into four uniform quadrants. A binary triangle tree structure (or bintree) use the same strategy but subdivide the initial rectangular region into two triangles, which are recursively subdivided in two halves. The main advantages of bintrees over quadtrees are that it's easier to work with LOD transition artifacts.

Multi-triangulation (Puppo, 1998) is an extremely general TIN data structure which stores a base mesh and the update operation required to refine it. Dependencies between update operations are represented by a direct acyclic graph which influence when simplification or refinement is performed.

Another data structure commonly used is the *pyramid structure* (used by TerraVision (Leclerc, et al., 1995 and Reddy, et al. 1999)), which is an adaptation of the texture *mipmap* technique (Williams, 1983) for geometry. Different LODs of a terrain geometry or texture are stored as different levels of a pyramid; the highest LOD at the bottom of the pyramid and each subsequent LOD is half the resolution of the previous.

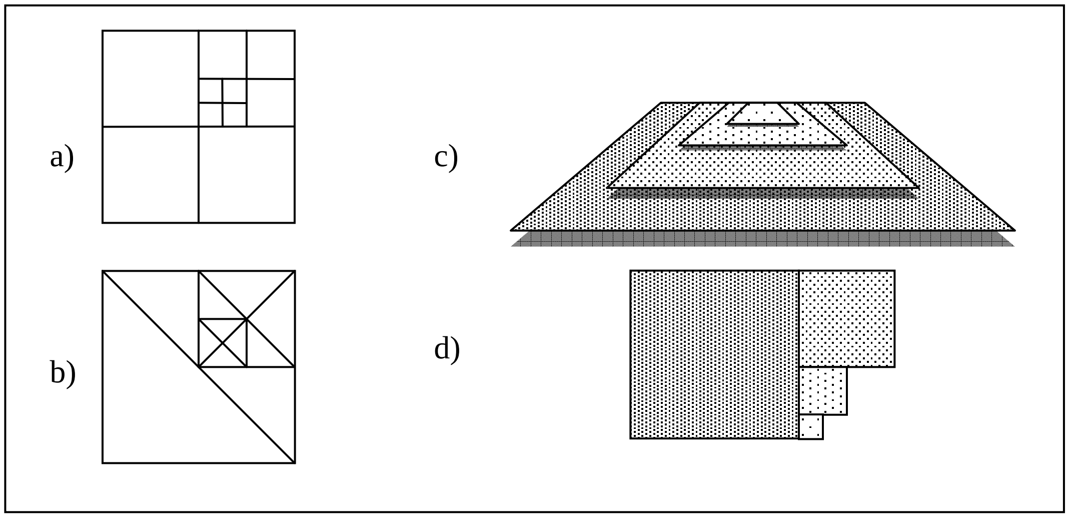

Fig. 2: Image (a) presents a quadtree of three levels. Image (b) presents a bintree of four levels. Images (c) and (d) present alternate representations of a pyramidal structure where each pattern represents a different resolution.

2.6 LOD selection

Since the basic theory states that less detail is required for small or distant elements of the scene, one of the most important aspects in a LOD algorithm is the criteria used to characterize an element as being small or distant , and how that criteria is used to select an appropriate LOD at runtime. The simplest criteria is distance, if an object is at a specific distance of the view position, it will be rendered using a particular LOD. An alternative criteria is the size of the particular element when projected into the screen, which in general is a more accurate way of selecting the LOD, but which is also more costly in terms of computation. Other parameters used in LOD selection may include view orientation and/or surface roughness. A more sophisticated approach proposed in (Zach, et al. 2002) involves defining benefit and cost functions for each LOD type (discrete or continuous) and based the selection on the benefit to rendering cost ratio.

3. Considerations

3.1 Terrain temporal discontinuity: the popping effect

The most common problem faced by all LOD techniques is minimizing the temporal discontinuity that occurs as geometric complexity suddenly changes when a LOD transition occurs. This effect is commonly referred to as the *popping effect* since it is particularly evident when a transition from a lower to a higher LOD causes more polygons to suddenly pop into view. The reverse effect is also common, as surface detail abruptly disappear when a higher to lower LOD transition occurs. The two most common approaches to this problem are examples of the inherent trade-off between performance and complexity present in LOD techniques. The popping effect can be easily eliminated by a) geomorphing (geometrical interpolation) between the two LODs as the view changes, maintaining visual continuity at the cost of the extra computation required to do so, or b) moving the LOD transition threshold farther away so that any potentially abrupt geometry change will occur before they can be perceived, due to the perspective projection. While this solution does not require extra computation, it will require maintaining geometric complexity beyond the point where it is perceivable, which goes against the basic premise of LOD.

3.2 Terrain spatial discontinuity: surface cracks and tears

A problem generally faced when using quadtrees or any tile-based LOD approaches is that it is possible to introduce artifacts such as *cracks* and *tears* in the edges between LODs caused when a polygon from a higher LOD does not share a vertex or lies on an edge of a polygon in a lower LOD. Common strategies to eliminate this kind of artifact include: modifying the polygon involved by either adding a vertex at the lower LOD triangle or adjusting the vertex of the higher LOD triangle, introducing new polygons to fill the gap or subdivided both polygon to produce a more continuous transition at the cost of adding geometrical complexity, or just preventing simplification of vertices that lie on the LOD boundaries.

3.3 Frame-to-frame coherence

In general, continuous LODs algorithms try to minimize as much as possible the computation required during a simplification or refinement process. Fortunately, for interactive applications where no drastic change in the view position or orientation occurs, it is possible to exploits the fact that for a particular region of the terrain, the LOD required during the next frame of animation will be either slightly lower (when moving away) or slightly larger when moving forward. This is known as *frame-to-frame or temporal coherence*. For a LOD algorithm to take advantage of frame-to-frame coherence, it must be capable of encoding the update operations (and the dependencies between each operation) in order to allow incremental updates to the last LOD instead of recomputing the new LOD from scratch.

3.4 Out-of-core operation

One of the main goals of terrain LOD is *out-of-core* operation: to be able to render terrain data sets that exceed the size of the available memory. By implementing a paging mechanism, it is possible to operate in memory only with the subset of the terrain data required for the desired LOD at a given time. As the view position changes, data is paged in memory as required. Since, by definition, out-of-core LOD algorithms do not require the complete terrain data to be loaded in memory, they are generally easier to adapt to use data streamed over the Internet.

Most out-of-core LOD algorithms work on blocks or tiles of terrains which are loaded when required. One of the simplest approaches is to exploit OS based system calls for mapping disk files to memory (memory mapping) in order to access terrain data. This approach delegates all the paging control to the OS, which is presumably more robust and efficient, but requires optimizing the terrain data on-disk layout (usually in coarse-to-fine order) and/or devising a efficient algorithm to map terrain coordinates to the vertex locations on disk. Other approaches depend on maintaining a mapping of nodes from a quadtrees or bintrees, or levels in a pyramidal data structure, to terrain tiles on disk. In (Bao and Pajarola, 2003) a specialized clustering technique for out-of-core LOD rendering is presented. The main improvement over previous techniques is the incorporation of space location and LOD constrained access patterns into the clustering algorithm and into the data structure design for a more efficient mapping of multi-resolution terrain data to external memory.

3.5 Texture management

Working with texture imagery for terrain rendering presents the same basic problem of data size and complexity faced with geometric data. In order to manage textures larger than memory, a texture paging mechanism is required to subdivide and/or downsample textures. Again, the same issues arise when combining texture tiles of different LODs. Fortunately, considerable work has been done in texture management techniques, such as texture caching, pre-fetching, mipmapping, and texture compression. Furthermore, support for some of these techniques can be found in commercial graphic hardware.

3.6 Geometry and texture detail synthesis

Even though the main goal of LOD algorithms is to reduce complexity to manageable size, often it is useful to able to introduce more detail than the available. Particularly, when combining low resolution textures with high resolution geometry, to cover discontinuities caused when combining geometry and/or textures from sources with considerable LOD differences, or to increase the realism of the terrain geometry or texture when the view position is very close to the terrain and there is no higher LOD available. The simplest solutions is to synthesize detail by blending the available data (geometric or texture) with a high-frequency fractal detail generated from a appropriate texture image.

3.7 Hardware supported algorithms

Current Graphic Processing Units (GPU) provide flexible programmable capabilities for graphics rendering which allow more control over the rendering pipeline than what was available using high-level graphic APIs. As consequence, new LOD schemes have been proposed in which the programmability of the GPU is exploited. GPU based LOD schemes range from off-loading part of the LOD computations to the GPU, adapting existing LOD techniques for GPU-based implementations, or devising new GPU-based LOD techniques specifically tailored around the strengths and limitations of the available GPU architectures.

4. Survey of LOD algorithms:

In this survey we review both the so-called classic LOD techniques and more recent techniques. In the case of techniques that present considerable differences from previous work, will focus on the basic description of the algorithms, data structures and handling of special considerations; when the technique represents an improvement over a previous techniques, we will summarize the major contribution presented.

4.1 Classic LOD techniques

Lindstrom et al. (Lindstrom et al., 1996) presented one of the earliest real-time continuous LOD algorithms. This technique is based on height fields. During an off-line operation, the original mesh is broken into a quadtree of blocks which contain discrete LODs. At runtime, an incremental top-down (coarse to fine) refinement is performed by traversing down the quadtree, followed by bottom-up simplification at the block level until the screen-space error criteria is reached. To exploit temporal coherence, an active cut of blocks is used to keep track of the current LOD. Vertex dependencies are used to prevent the introduction of cracks, but no re-meshing is performed between blocks nor any explicit geomorphing when switching from the simplest mesh of a higher LOD to the base mesh of the next lower LOD. Röttger et al. (Röttger et al., 1998) improved over the work of Lindstrom et al. by incorporating terrain roughness into the error metric for LOD selection and by explicitly supports geomorphing for smooth LOD transitions.

A particularly popular LOD algorithm call ROAM (Duchaineau, et al., 1997), employs two ordered priority queues; one for split operations and the other for merge operations. Split or merge is performed depending on the geometric error based on the triangle project size. Frame-to-frame coherence is afforded by the use of the two queues, where the level of detail progression can continue by traversing one queue, or reverse by traversing the other queue. Re-meshing is used to eliminate cracks, and geomorphing is performed to eliminate the popping effect.

Extending his previous work on progressive meshes (Hoppe, 1996 and Hoppe, 1997), Hoppe presented a TIN-based, view dependent terrain LOD algorithm (Hoppe, 1998). In this technique, continuous LOD is produce by adding or removing triangles from off-line generated blocks of terrain. LOD selection is done based on view frustum, surface orientation and screen-space geometric error. To eliminate the possibility of cracks, the algorithm prevents simplification of vertex at the block boundaries. This technique use memory mapping for out-of-core support.

4.2 Modern techniques

Thanks to the development of the consumer-level GPUs (originally introduced in 1999) there has been a huge increment in the graphic rendering capabilities available in common computers. Present GPUs throughput has surpassed 100M triangles/sec. Recent work acknowledges the need to reevaluate the problem of LOD rendering to better exploit the available rendering power of GPUs.

Starting with the premise that it is no longer necessary to find the ideal level of detail, but instead, to maximize the graphic hardware utilization while minimizing the CPU overhead, (De Boer, 2000) present the Geometrical MipMapping technique. This technique adapts the texure mipmap technique for geometric data. It employ height fields blocks of different LODs which are stored on disk for out-of-core operation and quad-tree of bounding boxes for view-frustum culling and block selection. Upon block selection, the vertex data is read from disk. To minimize CPU calculations, LOD transition are selected based on minimum and maximum viewing distance which are pre-calculated with the worst-case camera angle (the camera angle from which geometric error is more evident). Cracks are eliminated by changing the connectivity of the boundary vertices at the higher LOD. To eliminate popping, LODs are geomorphed by trilinear filtering. The work in (Brodersen, 2005) extends the GeoMipMap technique to support large-terrain textures by using a 3 level structure where the bottom level is composed of GeoMipMaps, the middle level consists of MapBlocks, which control the texture mapping for GeoMipMaps under it, and top level root node which represents the total terrain.

CABTT (Levenberg, 2002) extends the bintree-based LOD approach to work with clusters of aggregated triangles. This reduces CPU overhead by performing view culling per cluster. In addition, since clusters stay fixed over several frames, they may be cached on the video card memory. Similarly, BDAM (Cignoni et al., 2003) extends ROAM using a cluster of triangle called surface patches as basic unit. In addition, it integrates geometry and texture management into the same LOD algorithm by using a tiled quadtree for texture LOD management. QuickVDR (Yoon, et al., 2005) provide a general approach based on a cluster hierarchy of progressive meshes (CHPM) where each level of the hierarchy tree represents a different LOD. The hierarchy structure is used for visibility culling and each node or cluster consists of a progressive mesh which can be refined as required. The cluster itself is composed of a few thousand triangles with an associated bounding box which is built off-line. At runtime, clusters are split or merge as required. Popping effect is eliminating by requiring that the union of the full meshes of all child clusters equals the base mesh of the parent cluster.

Zhu (Zhu, 2005) presents a hybrid technique where irregular meshing is used to construct an input mesh for a uniform simplification process. This technique exploits the flexibility of an irregular mesh to produce better approximations of the original mesh, and the ability of uniform simplification to produce regularly-connected meshes which are ideal for creating triangle patches optimized for graphic hardware processing.

Cohen et al. (Cohen, et al., 2003) proposed a different approach with GLOD, which implements LOD management at the graphic library level. This is accomplished by extending the OpenGL API to support batches, objects, and groups. A patch corresponds to a vertex cluster which is submitted at full resolution but which may be rendered at different LOD. An object is a collection of patches which are simplified together to prevent cracks. A group is a collection of objects which share and adaptation mode.

Taking full advantage of the processing capacities available in current GPUs, the Geometry Clipmap technique (Losasso and Hoppe, 2004), reuse the texture mipmap approach but with a different GPU-based implementation. The basic algorithm works by building a multi-resolution mesh from the combination of nested concentric grids center about the view position, each grid from different level of LOD pyramid and sorted by decreasing resolution. As the view position changes, each nested grid is shifted to maintain the terrain LOD uniformly distributed around the view position and new data is paged into memory to fill the update region where LOD transition occurs. This technique integrates geometry and texture LOD and also supports terrain compression and synthesis.

A different GPU-based approach is presented in (Brodersen, et al., 2006), based on the progressive streaming of discrete mesh elements to the GPU. A nested mesh hierarchy of discrete LODs is built off-line. At run-time, view frustum culling is performed on the CPU and tiles identified as visible are sent to the GPU where they are interpolated in order to obtain a continuous LOD. A specialized memory manager is used to keep track of which data is on the GPU memory and is used to prevent unnecessary data transfer.

5. Visual Terrain Explorer

We are currently implementing a terrain visualization tool called Visual Terrain Explorer (VTE). Our long term goal is to provide an integrated visualization system for environmental monitoring applications which combine diverse data acquired through remote sensing techniques. In this visualization system we are exploring potential approaches for combining the visualization of spatial-temporal data with hydrological modeling, and applying them to study the Jobos Bay Reserve area.

Based on a modular software design, we have developed a working prototype capable of rendering digital elevation maps of various formats with associated texture images. Various experimental LOD capabilities are currently being integrated into the prototype based on the results of the survey presented here. Our selection of LOD technique will be described in the next section.

Beyond LOD management, we plan to research more formal specialized techniques for out-of-core data management in order to define and implement the components required for remote data acquisition.

6. Conclusions

Based on the algorithms reviewed we can identify a series of common trends in modern LOD algorithms. The most significant trends are the use of triangle patches as basic rendering unit, the de-emphasis of global optimization (i.e. obtaining the perfect level of detail), and the emphasis on out-of-core operation as a required aspect of level of detail management.

The most important advantage of using triangle patches as basic rendering unit is that they allow maximizing the rendering capabilities of current GPUs. A secondary advantage is that their use allows for a better coupling between the geometry and the texture LOD techniques, simplifying the development of unified geometry and texture LOD techniques. The use of triangle patches also fit appropriately with out-of-core operation.

Generating triangle patches optimized for GPU processing is generally incompatible with producing a globally optimized mesh, which is usually irregular in detail distribution. Thus, many new LOD techniques relax the optimization goal at the global level and focus in optimizing at the local (triangle patch) level. The positive side effect is that the resulting techniques are in general simpler and require less CPU processing than previous techniques.

Modern LOD techniques recognized out-of-core operation as a required aspect of LOD management. Traditionally LOD management was concerned with being able to render complex geometry interactively, now the definition has expanded to incorporate scalability. A LOD technique must be able to support interactive rendering independently of the data size.

References

Clark, J. H. (1976). “Hierarchical geometric models for visible surface algorithms”. *Commun. ACM* Vol. 19, No. 10, pp 547-554.

Luebke, D., Reddy, M., Cohen, J., Varshney, A., Watson, B., and Huebner, R. (2003). *Level of Detail for 3D Graphics*, Morgan-Kaufmann, Los Altos, CA.

Fowler, R. J. and Little, J. J. 1979. (1979). “Automatic extraction of Irregular Network digital terrain models”. *Proceedings of the 6th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '79. ACM Press, New York, NY, pp 199-207.

Samet, H. (1984). “The Quadtree and Related Hierarchical Data Structures”. *ACM Comput. Surv*. Vol. 16, No. 2, pp 187-260.

Puppo, E.(1998), “Variable resolution triangulations”, *Computational Geometry*, Vol. 11, No. 3-4, pp.219-238.

Leclerc, Y. G. and Lau, S. Q. (1995). “TerraVision: A Terrain Visualization System”, Technical Note 540. AI Center, SRI International, http://www.ai.sri.com/pubs/files/778.pdf 02/15/2006

Reddy,M., Leclerc,Y., Iverson,L., and Bletter, N. (1998) "TerraVision II: Visualizing Massive Terrain Databases in VRML," *IEEE Computer Graphics and Applications*, Vol.19, No.2, pp. 30-38.

Williams, L. (1983). “Pyramidal parametrics”. *Proceedings of the 10th Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '83. ACM Press, New York, NY, 1-11.

Zach, C., Mantler, S., and Karner, K. (2002). “Time-critical rendering of discrete and continuous levels of detail”. *VRST ’02: Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 1–8.

X. Bao and R. Pajarola, (2003) “Lod-based clustering techniques for efficient large-scale terrain storage and visualization,” *Proceedings of the SPIE - The International Society for Optical Engineering*, Vol. 5009, pp. 225–35.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. (1996). “Real-time, continuous level of detail rendering of height fields”. In *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '96. ACM Press, New York, NY, 109-118.

Röttger, S., Heidrich, W., Slussallek, P., and Seidel, H. P. (1998) “Real-Time Generation of Continuous Levels of Detail for Height Fields”. *Proceedings of the 6th International Conference in Central Europe on Computer Graphics and Visualization*, 315--322.

Duchaineau, M., Wolinsky, M., Sigeti, D. E., Miller, M. C., Aldrich, C., and Mineev-Weinstein, M. B. (1997). “ROAMing terrain: real-time optimally adapting meshes”. *Proceedings of the 8th Conference on Visualization '97*. R. Yagel and H. Hagen, Eds. IEEE Visualization. IEEE Computer Society Press, Los Alamitos, CA, 81-88.

Hoppe, H. (1996). “Progressive meshes”. *Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques* SIGGRAPH '96. ACM Press, New York, NY, 99-108.

Hoppe, H. (1997). “View-dependent refinement of progressive meshes”. *Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques* International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, pp. 189-198.

Hoppe, H. (1998). “Smooth view-dependent level-of-detail control and its application to terrain rendering”. *Proceedings of the Conference on Visualization '98*. IEEE Computer Society Press, Los Alamitos, CA, pp 35-42.

De Boer, W. H., (2000). “Fast terrain rendering using geometrical mipmapping”. http://www.flipcode.com/articles/article-geomipmaps.pdf. 02/15/2006

Brodersen, A. (2005). “Real-time visualization of large textured terrains,” *GRAPHITE ’05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and SouthEast*

Asia, pp. 439–442.

Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., and Scopigno, R. (2003). “BDAM - batched dynamic adaptive meshes for high performance terrain visualization,” *Computer Graphics Forum*, Vol. 22 No. 3 pp505-514.

Levenberg, J. (2002) “Fast view-dependent level-of-detail rendering using cached geometry”. *VIS ’02: Proceedings of the conference on Visualization ’02*.

Yoon, S., Salomon, B., Gayle, R., and Manocha, D. (2005). "Quick-VDR: Out-of-Core View-Dependent Rendering of Gigantic Models," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, No. 4, pp. 369-382.

Zhu. Y (2005), “Uniform Remeshing with an Adaptive Domain: A New Scheme for View-Dependent Level-of-Detail Rendering of Meshes”. *IEEE Transactions on Visualization and Computer Graphics* Vol. 11, No. 3, pp. 306-316.

Cohen, J., Luebke, D., Duca, N., and Schubert, B. (2003) “GLOD: A geometric level of detail system at the OpenGL API level,” *VIS ’03: Proceedings of the 14th IEEE Visualization 2003 (VIS’03)*, pp. 85.

Losasso, F. and Hoppe, H. (2004). “Geometry clipmaps: terrain rendering using nested regular grids”. *ACM Trans. Graph.* Vol. 23, No. 3 , pp. 769-776.

Schneider J., and Westermann, R. (2006) “Gpu-friendly high-quality terrain rendering,” *Journal of WSCG*, Vol. 14, No. 1-3, pp. 49–56.

Authorization and Disclaimer