QLOD: A Data Structure for Interative Terrain Visualization

RODRIGO TOLEDO[1], MARCELO GATTASS[1], LUIZ VELHO[2]

[1]Tecgraf - Computer Graphics Technology Group, Computer Science Department, PUC-Rio
{rtoledo,gattass}@tecgraf.puc-rio.br

[2]IMPA - Institute of Pure and Applied Mathematics
lvelho@visgraf.impa.br

Abstract. This work focuses on the use of hierarchical structures to interactively visualize terrains. Our purpose is to achieve real time three-dimensional visualization of terrain models obtained from a geographic information system. These models can represent a broad area of the Earth's surface with detailed information, thus yielding very large datasets. For this reason, it is necessary to use a compatible structure to store and query this data. In this paper, we present a new data structure which combines advantages of different structures provided in the literature. We also discuss an implementation of these ideas and a series of tests, drawing several conclusions from them.

1 Introduction

Geographic data have been extremely important to the evolution of mankind. Maps have been used for centuries to describe geographical information. The development of techniques and equipment made data collection more precise and voluminous, thus contributing for the increase of activities in which the geographic data can be applied.

The size of the areas covered by maps can vary considerably. Maps can describe everything from the coast of a continent to a highway crossing in a small town. To handle such situations, the geographic representations must provide information on scale, so that one can know the proportion between represented distances and real ones. The use of different scales provides a flexibility which serves various needs concerning levels of detail according to what one wishes to represent and to the area described by the map.

With the emergence of Computer Science, geographic data reached digital format, thus providing dynamism to their applications. However, the development of computational tools capable of treating such complex information is necessary, as data sets can be huge. Furthermore, as visualization is concerned, depending on the observer's point of view there are diverse needs regarding levels of detail for different parts of the terrain, giving more emphasis (higher resolution) to what is being closely seen in a particular moment. Among the several strategies used to face such challenges, multiresolution stands out for following the natural idea of working with different scales. The goal is to obtain a structure capable of storing all of a terrain's data and allowing a quick (either partial or complete) extraction of the information at the desired scale.

The present work focuses on a computational framework for treating geographic data and obtaining such extraction efficiently, so that terrain visualization can be performed in real time. The solution proposed is a combination of already known strategies, such as progressive mesh and multiresolution structures, comprising their advantages and reaching efficient results.

2 Related Work

In order to obtain real-time mesh extraction according to visualization parameters, in a preprocessing step, multiresolution structure must be constructed. Such structure must be capable of making fast extractions of an adapted representation.

This type of solution has been the subject of recent research. The studies are not limited to meshes which represent terrain, but also to 3D generic meshes, as there is always interest in adapting a mesh resolution to the needs of interactive visualization.

Below we summarize previous work in this area. They are divided in two groups: those that use irregular meshes and those that use regular meshes.

Irregular Meshes:

- The work on terrain by Hughes Hoppe [8] extends to his previous works. It proposes a hierarchical construction of the progressive mesh [7] in which the terrain is divided in parts. Hoppe's work differs from the present article especially for having only one progressive mesh for the whole terrain.
- The work by Carl Erickson and Dinesh Manocha [4] was based on the work by Michael Garland and Paul Heckbert [6]. A new structure, called HLOD (Hierarchical LOD[1]), is defined and used for several objects

[1]LOD: Level Of Detail

in a virtual scene. This structure is based on a tree in which each object in the scene is a node and has its various levels of detail stored. In this case, the terrain is partitioned and treated as a group of three-dimensional objects, without taking any advantage of the fact that it is a height field.

Other important works which also used irregular meshes are those by Puppo et al [1] and by Mark de Berg [2]. They employ a multiresolution Delaunay triangulation.

Regular Meshes:

- In the work by Lindstrom et al [9], the variable-resolution structure was based on regular meshes. It always uses a regular grid with the height information as initial data. Each triangle can be divided in two, and the decision of whether they are to be divided or not depends on their vertical difference (error). This difference is projected on the screen so that it can be confronted against the error margin specified by the user in the image space. There is a dependency rule for the division operation in order to avoid gaps in the terrain's mesh.

- In *ROAMing Terrain* [3], the mesh is treated as a binary structure in which the triangles can be divided or grouped two by two by *split* and *merge* operations. In this work, the goal was to always draw the same amount of triangles at each redrawing, taking advantage of the last draw's triangulation.

Other important works using regular mashes are those by Topo Vista [5] and by Pajarola [11]. They use a restricted and triangulated Quadtree structure.

The structure proposed in the present article combines both mesh concepts, attempting to avoid problems related to each one while maintaining their qualities.

3 The QLOD structure

QLOD is a variable-resolution structure which supports irregular triangulation and has an underlying regular structure. It aims at combining the characteristics of progressive irregular meshes and multiresolution regular meshes.

QLOD is based on an irregular mesh decomposed into regular blocks. A regular block is an irregular mesh whose domain is a regular region. The interior of a regular block is always an irregular mesh which can be adapted to the terrain's characteristics. However, the boundary of the regular block is composed by four straight polygonals, forming a rectangle. Each polygonal is composed by the edges of the irregular mesh (Figure 1).

Each regular block is stored as a progressive structure [7]. Its resolution can vary between the final mesh and the base mesh (the variation is determined by the number of vertices). The progressive structure used in QLOD works with boundary restriction, that is, the boundary is the same for all resolutions, therefore the minimum resolution always includes the original boundary of the regular block (Figure 2).

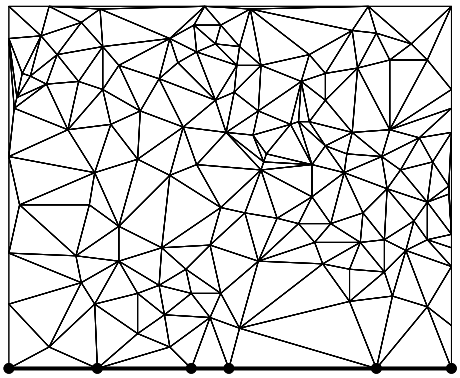

Figure 1: The regular block is an irregular mesh whose domain is a regular region (the highlighted inferior boundary is a straight polygonal).

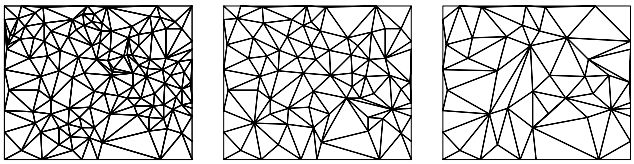

Figure 2: Progressive mesh structure with boundary restriction. Mesh in three different resolutions, with 129, 80 and 50 vertices.

With the QLOD structure we can have, at the same time, an irregular mesh and a regular subdivision. This is so because the irregular mesh is totally covered by regular blocks. Two neighboring regular blocks have a boundary in common, whose composing edges are the same for both. The union of all boundaries in these blocks constitutes the underlying regular structure (Figure 3). The regular structure is underlying because its edges are not explicit in the mesh, but only through straight polygonals.

From the underlying regular structure it is possible to make a hierarchical decomposition of the terrain's domain. The whole terrain can be seen as a single regular block which, by means of a recursive subdivision process, generates several regular blocks at different hierarchical levels. The underlying regular structure then becomes a multiresolution structure in which each hierarchical level completely covers the terrain's domain, and each lower level is always a subdivision of its upper level.

QLOD forms a hierarchical decomposition of the domain, since the regular blocks can be merged four by four, constituting a new regular block one level above in the hi-

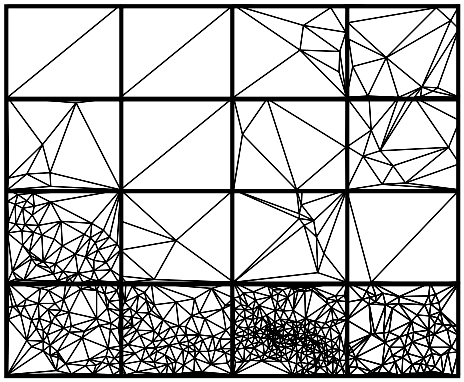

Figure 3: Irregular mesh with underlying regular structure.

erarchy. The hierarchy can be represented by a quaternary tree in which each node is a regular block containing an irregular mesh of triangles. Each level in the tree represents a hierarchical level of the regular structure. This means that a given node covers the same area as its sons (Figure 4) and that, at each level, all the nodes form a different regular decomposition of the domain.

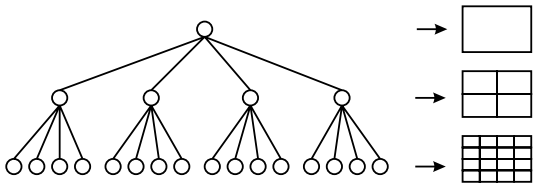

Figure 4: Quaternary tree representing the multiresolution regular structure.

For the hierarchical decomposition to represent the multiresolution mesh, the terrain representation in each upper level in the tree must be simpler than in the level below. This means that at each level the same area of the domain is represented by a different number of polygons, vertices and edges (fewer in the upper levels). A father node is actually the union of son nodes' simplified meshes (Figure 5).

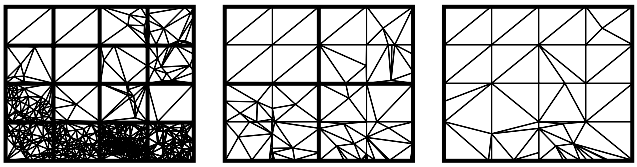

Figure 5: Multiresolution mesh. Each father node is the union of its sons' base meshes.

Each regular block can continually have its resolution modified, in a progressive way (Figure 2). Such variation independs from the other blocks, because the boundary restriction does not allow the operations in one block to interfere in another block. Since the resolution can be locally modified at any point of the domain, QLOD is a variable-resolution structure.

The underlying regular structure can also be either refined or simplified, and such operations are also independent in each block. This unrestricted hierarchical structure is possible due to the boundary restriction.

Even though the boundary restriction provides the above mentioned benefits, it also has some drawbacks. When the interior of a block is simplified, its boundary remains unaltered, so that the resolution of the interior and the boundary is very different, being much greater in the last one. In the higher hierarchical nodes this difference implies a quality loss in the mesh's triangles, which can have a bad aspect ratio. Nevertheless, this only occurs more emphatically on the terrain's surroundings and in situations in which resolution is very low (for instance, when the observer is far away from this portion).

4 QLOD Construction

For building the QLOD, the first step is preprocessing. Its purpose is to adapt all input data of a terrain to the necessary format in the QLOD building algorithm - that is, this step depends on the format of the original data. The terrains used as examples in this work were described by heights maps. The desired output format of this stage is an irregular triangulation with an underlying regular structure (Figure 3).

A natural way to construct the QLOD is to use simplification algorithms to build a progressive mesh, and to use a Quadtree construction algorithm. These algorithms must work together for the goal to be achieved: an irregular mesh with an underlying regular structure in multiresolution.

Hierarchy Construction is the first processing step. The input format is an irregular triangulation with an underlying regular structure. Whatever the preprocessing is, it must result in this format so that the mesh's hierarchy can be built. Furthermore, the number of regular blocks (in the underlying regular structure) must be a power of two in both directions of the domain.

The resulting hierarchical mesh is represented by a Quadtree. The Quadtree is built from the leaves to the root. The blocks are put together four by four, constituting the higher level block. This operation must occur at each level, so that the whole domain is covered. This process is repeated until a single block has been built.

Each initial block already has its triangulation (from the preprocessing stage), and each block constructed in this step also has its triangulation, generated from the union of the son nodes' triangles. Thus, every node in the Quadtree will have a triangulation.

At the end of this step, a hierarchical decomposition is

obtained in which each hierarchical level is the union of the lower levels.

Simplification of the Regular Blocks is the second processing step. The structure resulting from the previous step is a hierarchical decomposition of the terrain's domain, but all levels have the same resolution, because no simplification was made to the triangulation. Now the purpose of this step is to obtain a multiresolution structure.

For the resolution change to be possible, a progressive mesh [7] is to be created from the triangulation in each regular block. This is done starting from the leaf nodes of the Quadtree. The construction of a progressive mesh is made by means of simplification operations following a predetermined criterion. The progressive structure employed here was MLOD [10], which uses the edge collapse operation.

In the QLOD, two characteristics are important for constructing the progressive mesh:

- Boundary restriction - As was already seen, it is essential that the boundary is not altered during simplification, so that it coincides with the neighboring regular block. This also implies determining that the base mesh must contain at least the boundary vertices.
- Simplification level control - This control can be numerically represented by the percentage of remaining vertices after each simplification iteration. It is especially important in two moments: during visualization (as will be seen on the next section), to determine each regular block's resolution; and on the construction stage itself, to determine which is the simplification level of the base mesh.

As well as in the leaf nodes, the progressive mesh is also built in the other nodes of the Quadtree. Nevertheless, the final (more refined) mesh in each node is the union of the base meshes of each son node. In fact, the previous step, hierarchy construction, is performed together with this simplification stage. Since the simplification of a node depends on the previous simplification of its sons, the construction occurs from the leaf nodes to the root (bottom-up).

Each block has a final mesh which is simpler than the union of the son nodes' meshes; therefore the Quadtree hierarchy becomes multiresolution. At each higher level in the hierarchy, the mesh becomes simpler (Figure 5).

5 Visualization

Visualization is always limited to the image resolution, which is fixed and determined by the number of pixels. A natural solution to increase efficiency is to adapt the terrain's resolution to that of the screen, thus avoiding the excess of triangles drawn and fulfilling interactive visualization requirements. The resolution degree varies according to the terrain's projection on the screen. When a portion is being closely viewed, the resolution must be much higher than for far portions. Considering that generally a terrain is viewed inclined (the observer is standing on the terrain looking towards the horizon), the extracted resolution can continually vary on the surface.

The adaptation function is what defines which will the resolution be along the mesh at the moment of the visualization, according to some predetermined criteria. Several criteria can be applied, taking camera parameters into account. An example of a such criterion is a distance-based adaptive function (Figure 6).

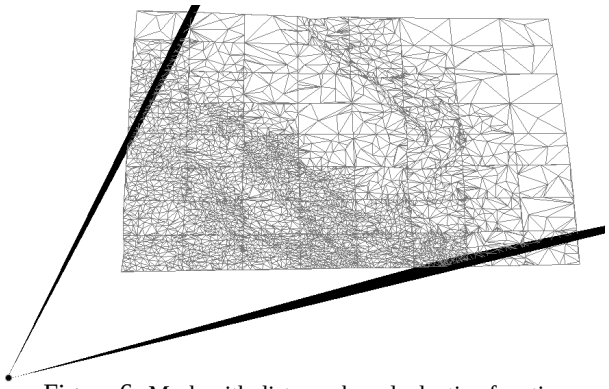

Figure 6: Mesh with distance-based adaptive function.

The visualization algorithm has the following steps:

```
VISUALIZE()
  Read structure representing the terrain
  Forever
    Obtain camera parameters
    EXTRACT_MESH (parameters)
    Draw extracted mesh
```

The QLOD representing the terrain is read first. A loop is performed, at each iteration the camera parameters are collected. From these parameters, an "optimum" mesh is extracted and drawn. Mesh extraction (`EXTRACT_MESH`) is a task in which efforts must be joined, as it will be directly responsible for obtaining a good image update rate, which will generate the feeling of an immediate feedback to interaction.

Mesh Extraction is done by means of the following algorithm:

```
EXTRACT_MESH (paramaters)
  Calculate an adaptation function
  DECIDE_MESH_RESOLUTION (adaptation function)
  Return mesh with calculated resolution
```

The metrics of the adaptation function implemented is based on the distance from the mesh to the observer. It is a simple heuristics which works very well (the visualization

algorithm and QLOD allow the use of more sophisticated adaptation functions). The portions of the terrain closer to the camera will be drawn more refined than those farther away. This rule applies along the whole domain. The mesh resulting from this function tends to be compatible to the screen resolution. This compatibility is greater because, without an adaptation function, portions distant from the projection screen would have a much greater resolution than that of the screen (thus drawing many triangles in a few pixels), but with this heuristics such portions are simplified, being more similar to the screen resolution.

The adaptation function implemented does not deal linearly with distance. Instead, it uses distance projected with perspective and normalized between values 0 and 1. This non-linearity yields good results for the adaptation function. Such value is also used by the Z-Buffer (implemented in graphics libraries) to compare depths. Therefore, the graphics library can be requested to perform the computations. The centroid of each regular block is used as a reference for the projection.

The `DECIDE_RESOLUTION_MESH` is performed in two steps: ideal resolution estimation and actual resolution estimation.

Ideal Resolution Estimation The algorithm begins by computing what would be the ideal mesh resolution for current camera parameters, traversing the QLOD structure. This step is divided in two tasks: (1) Selecting the hierarchical level inside the regular structure; (2) Selecting the intra-hierarchical resolution.

The first task is performed by traversing the QLOD hierarchical structure top-down. From the regular block of the father node, its distance to the projection screen is computed (adaptation function) and the need for refinement is checked. If this is the case, the algorithm recursively performs the same check for each son node of the lower hierarchical level. Recursion stops when one of the three following situations occurs: (1) A level was reached in which there is no need for further refinement; (2) There are no son nodes; or (3) The regular block is outside the viewing frustum. As a result, a cut in the hierarchical quaternary tree is obtained, and the regular blocks to be drawn will be those on the leaves of the sub-tree above the cut (Figure 7). The top-down strategy is essential for the speed of the algorithm.

The second task decides, for each regular block selected in the previous task, what will the progressive mesh resolution be. All progressive meshes have a minimum resolution (base mesh) and a maximum resolution (final mesh), and allow a great granularity of intermediate meshes. One of these meshes is to be chosen, according to the adaptation function.

The division of this step in two tasks is related to QLOD's dual characteristic. In the first task, the regular structure's

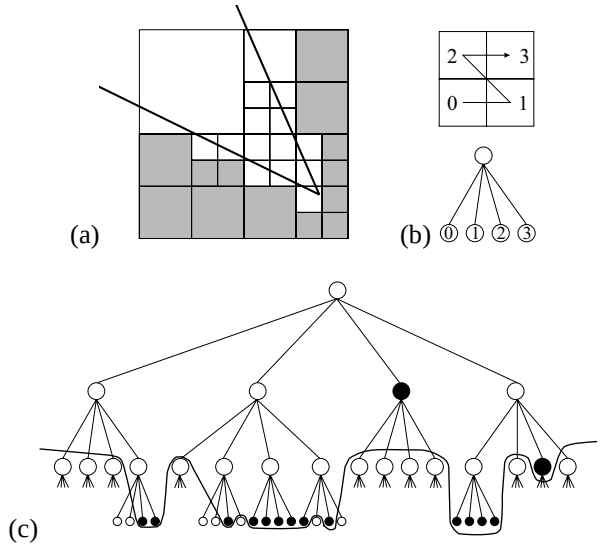

Figure 7: (a) QLOD with regular blocks in different refinement levels inside the regular structure. The region between the lines traversing the QLOD represents the viewing frustum, and the vertex represents the observer. Regular blocks outside the viewing frustum are gray. (b) Order in which the regular blocks are to be represented in the Quadtree. (c) Cut in the Quadtree. The subtree's leaf nodes (in black) above the cut are to be drawn.

resolution is being defined, while the second task deals with the irregular mesh's resolution. These tasks can also be compared to the selection of a value for a real number, in which one first calculates the integer part and then the fractional part.

Actual Resolution Estimation As we have seen, at each frame the ideal resolution is estimated, but it can be different from the one drawn in the previous frame. In this case, if the ideal resolution is drawn, there will be a noticeable pop, which one should avoid. To deal with this, a mesh resolution is defined as being a transition between the last frame's mesh and the ideal mesh. Therefore, it is necessary to store the resolution of the last drawn mesh.

During visualization, a continuous transition will always be happening between what is actually being drawn and the ideal resolution. The transition speed accords to the resolution change needs. At each frame, the transition mesh is estimated, whose resolution will have an increase rate proportional to that of the ideal resolution. If the speed of the camera increases, then both rates are increased as well.

The transition mesh is computed based on the position of the progressive mesh's vertices. For example, if the ideal mesh has a smaller resolution than the last drawn mesh and one wishes to make a transition between both, the edges to be collapsed have their size slowly decreased, until they have completely disappeared. This is done by making a

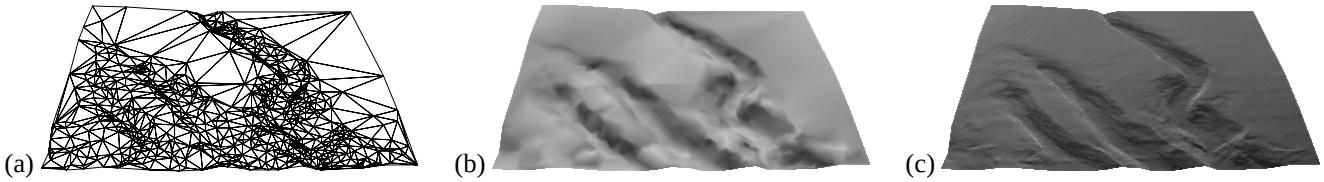

Figure 8: (a) Wireframe of a mesh. (b) Image illuminated in real time. (c) Mesh with same geometry but with pre-illuminated texture.

smooth interpolation of the position of the vertex that will disappear together with the edge between its position and that of the edge's other vertex (Figure 9). To increase the resolution, the interpolation is inverted. This procedure can only be performed when the resolution change is within the same hierarchical level (in each regular block). When there is the need to change the resolution between different hierarchical levels, this proceeding is divided in two steps: first the interpolation is performed until a less refined mesh is obtained (in case one is decreasing the resolution) in the current hierarchical level; then, starting from the more refined mesh in the lower level, a new interpolation is done until the desired resolution is achieved. As was previously noted, the smallest mesh resolution in a given hierarchical level equals the greatest resolution of the lower level; that is, the level change is actually unnoticeable to the observer.

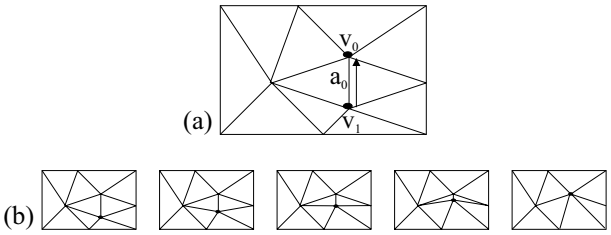

Figure 9: (a) The mesh will have its resolution decreased, and edge a_0 will be collapsed. (b) Vertex v_1 will have its position interpolated until it completely disappears.

Texture and Illumination A terrain's appearance can also be influenced by illumination. Even though current graphics libraries support illumination, it still implies some costs. First, the normals of each vertex must be stored (or computed in real time). At each frame, for each vertex and each light, the new vertex color is computed. Finally, the graphics library interpolates the vertex values inside the triangles (Gouraud Shading).

The use of pre-illuminated textures or even satellite images does not require the illumination computation. The image these textures represent already contains either an artificial (pre-illuminated textures) or natural (satellite pictures) shading. Applying textures by means of graphics libraries is much more efficient than computing illumination, because one only has to provide, for each vertex, the texture coordinate. For a terrain, it is the position of the vertex in the domain. Moreover, the appearance can be improved, because when illumination is applied the resulting shading is proportional to the mesh resolution (Figure 8b), while with textures one can work with a higher resolution (Figure 8c). For such reasons, we decided to use a pre-illuminated texture instead of computing the illumination on the fly.

6 Results

The results presented in this article were obtained from a path on a submerse terrain in the Brazilian coast (Figure 10)), running under a Pentium III 500 MHz with memory of 128 Mb and a Viper 770 video board. The terrain's area is not rectangular, but rather quite irregular, with a total of 60 million heights samples. The preprocessing stage in the QLOD construction lasted about 1:30 hour, generating files adding up to 25 Mb and a total of 725,000 triangles. The QLOD construction process lasted around 1:30 minute and generated 135,000 new triangles in the upper hierarchical levels. Together with the texture (which was artificially generated), the whole terrain occupied 90 Mb of RAM when loaded.

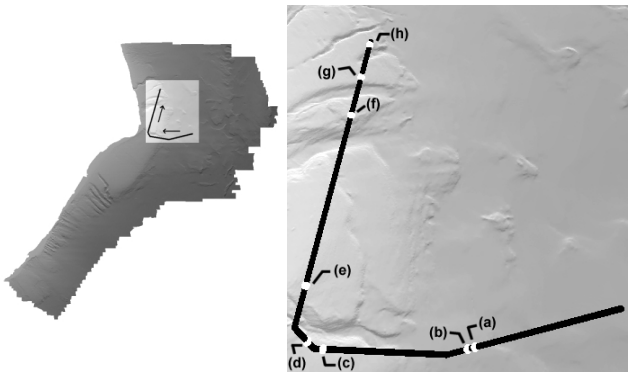

Figure 10: Submerse terrain used for tests.

Along this path, some important points are highlighted (Figure 10). The camera begins at a certain height and moves down until it reaches point (a); then up to point (c) it gets closer to the terrain's boundary, and there it makes a sharp curve right, going back to the inner part of the terrain. At point (g) the camera starts to turn down until it reaches the end of the path, being completely inclined. The total

time for the circuit was 30 seconds.

A total of 1169 frames were drawn continually the space of 30 seconds, with an average performance of 38.96 fps. Along the path, some results were obtained, as shown in the graphs in Figure 11.

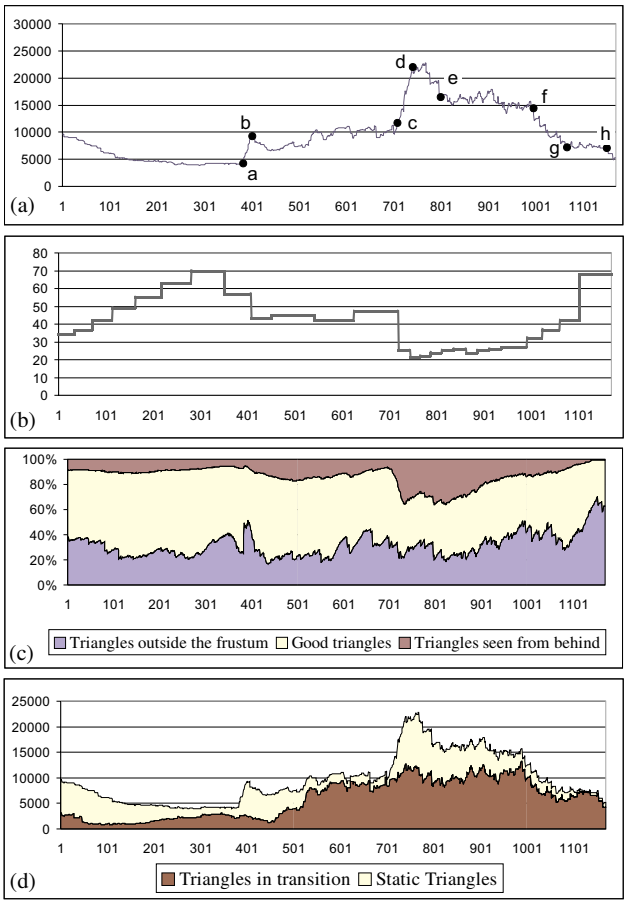

Figure 11: (a) Total primitives drawn along the frames. (b) Frame rate along the frames (in fps). (c) Distribution of the percentage of triangles drawn. (d) Amount of triangles in transition at each frame.

As was expected, the frame rate (Figure 11a) is inversely proportional to the amount of triangles drawn (Figure 11b). Between point (a) and point (b) there is an increase in the amount of triangles, from 4300 to 9300. This occurs because the camera, which was turned down, starts to point forward, thus visualizing a broader portion of the terrain. Between points (c) and (d), there is another increment in the amount of triangles because the camera, which was almost at the terrain's boundary, makes a sharp curve to the right, heading back to the inside of the terrain. Between points (e) and (f) the path is a straight line, and the amount of triangles does not vary much. From point (g) to point (h), the camera is directed at the same spot but keeps moving. As it is close to the terrain, the total amount of primitives is small, since what is outside the viewing frustum is not drawn.

In the graph in Figure 11c, the upper band indicates the percentage of triangles that were drawn despite being seen from behind, and the lower band shows the percentage of triangles outside the viewing frustum which were also sent to the graphics board. Both bands represent a waste of triangles, since such triangles do not appear in the image. In the band of triangles seen from behind, it is possible to notice an increase between frames 700 and 1000 (corresponding to around points (c) and (f)). This occurs because in this portion there is a series of small bumps in the terrain which are seen in an angle such that the triangles in the hidden part were inverted (with their normal in the same direction as the camera). At the end of the path, the amount of triangles seen from behind is almost null, as the terrain is viewed from above. The existence of triangles being drawn outside the viewing frustum is due to the fact that a given level in the QLOD must be either totally left out or totally drawn. In the latter case this can happen even if only a small part is being shown (Figure 6). The average waste was 32% of triangles drawn outside the frustum, and 16.5% of inverted triangles. Therefore, only 51.5% were actually valid. The total amount of triangles outside the frustum can be decreased with a greater refinement of the hierarchical subdivision. The same test was performed with an additional level on the QLOD hierarchy, and the average waste was of approximately 15% of triangles drawn outside the frustum. On the other hand, the overall performance decreased. The efficiency loss is due to a great extent to the hierarchy. As it becomes more complex, with several additional nodes, the visualization algorithm takes more time to extract the mesh to be drawn.

On the last graph (Figure 11d), the dark band represents the triangles in transition, while the light band indicates the static ones. It is interesting to notice that the light band increases in two moments: between points (a) and (b), and between points (c) and (d). In these portions there was a change in the direction of the camera, and new portions of the terrain were being viewed. In such cases an optimization is made, no longer requiring the transition of the portions which are just being made visible. Approximately after point (g) the band of static triangles almost disappears, as at the end of the path the camera is directed at the same spot, only getting closer to it; that is, there are no new portions of the terrain entering the viewing frustum.

At points (b) and (f), the projected areas of all triangles were computed and classified in three groups (measured in pixels): "smaller than 1", "from 1 to 20" and "greater than 20". Only the triangles inside the viewing frustum were measured (their total number is in parentheses in Figure 12), and the camera position was still, so there were no triangles in transition. For comparison purposes, the same

measures were taken for the triangles with maximum resolution, without any level of detail simplification or reduction. The results are presented in Figure 12.

Analyzing these results, one can notice the strong influence of QLOD in reducing the "smaller than 1" and "from 1 to 20" groups where there were very small triangles that could be simplified.

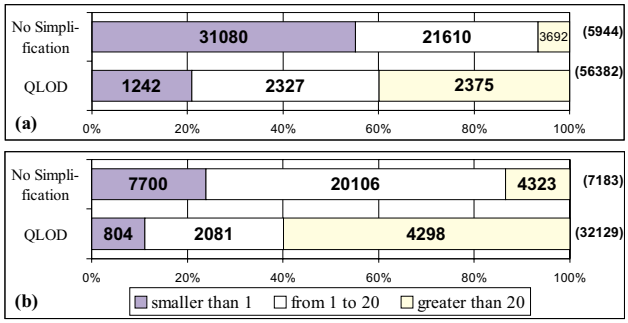

Figure 12: Distribution of drawn triangles according to the projected area. (a) At point b of the path. (b) At point f of the path.

7 Conclusion

With QLOD it was possible to interactively visualize complex terrains.

The division of the terrain in regular blocks allowed the extraction of meshes adapted to camera parameters, since each block can have a different level of detail.

The joint use of characteristics from regular and irregular meshes yielded good results. QLOD's underlying regular structure allowed a fast exclusion of portions of the terrain outside the viewing frustum. The irregular mesh allowed a better adaptation to the terrain, as well as the use of a progressive mesh, which allows a very fast extraction of meshes with different level of detail.

The use of Z-Buffer for visualization naturally allows other objects to be viewed together with the terrain.

The adaptive function based only on distance, despite being simple, was able to perform efficient reductions in the amount of triangles to be drawn.

8 Future Work

There can always be situations in which the whole terrain does not fit the memory, and access to secondary memory is inevitable. The use of QLOD could be extended for a situation in which the terrain can be loaded to the memory in real time.

Creating an adaptive function more sophisticated than the current one can yield even better results for the application.

We suggest a more effective analysis concerning the use of different level of detail for textures, making them totally independent from the mesh's level of detail.

A study can be made on solutions for decreasing the waste of triangles drawn outside the viewing frustum without a performance loss.

Finally, a valid work is to identify how the QLOD can be extended to generic surfaces, not only for terrains. Many definitions of the QLOD assume the existence of a domain in $\Re^2$ in which the terrain is defined, thus preventing a direct application for generic models.

References

[1] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997. ISSN 0178-2789.

[2] Mark de Berg and Katrin T. G. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60(1):1–12, January 1998.

[3] M. Duchaineau, M. Wolinsky, D. Sigeti, M. Miller, C. Aldrich, and M. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. *IEEE Visualization '97*, pages 81–88, November 1997.

[4] Carl Erickson and Dinesh Manocha. Simplification culling of static and dynamic scene graphs. 1997.

[5] W. Evans, D. Kirkpatrick, and G. Townsend. Right triangular irregular networks. Technical report, University of Arizona, 1997.

[6] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH 97*, pages 209–216, August 1997.

[7] Hugues Hoppe. Progressive meshes. *Proceedings of SIGGRAPH 96*, pages 99–108, August 1996.

[8] Hugues H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. *IEEE Visualization '98*, pages 35–42, October 1998.

[9] Peter Lindstrom, D. Koller, W. Ribarsky, L. Hughes, N. Faust, and G. Turner. Real-time, continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH 96*, pages 109–118, August 1996.

[10] Paulo Mattos. Mlod: Biblioteca para simplificação de malhas tridimensionais. Technical report, Pontifícia Universidade Católica (PUC-Rio), 1999. `www.tecgraf.puc-rio.br/~pmattos/mlod`.

[11] Renato B. Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. *IEEE Visualization '98*, pages 19–26, October 1998.