Multiresolution Ordered Meshes

J. Ribelles, M. Chover, J. Huerta, R. Quirós
Departamento de Informática. Universidad Jaume I.
Campus de Penyeta Roja, 12071. Castellón. Spain.
Tel: (964) 345771 / Fax: (964) 345848
e-mail: ribelles@inf.uji.es

Abstract

Geometric models of high complexity are rapidly becoming commonplace. These models are usually represented by means of complex triangle meshes that create three important problems: visualization, transmission and storage. Interest in multiresolution representation is very keen, because it is an efficient method for addressing these problems. We propose a new data structure, multiresolution ordered meshes (MOM), with low storage cost and an efficient algorithm to extract any level of detail. Meshes are stored in order depending on the simplification process. It allows to work interactively with highly detailed models and to transmit them progressively over a communication line. The model construction is very simple because the data structure is based on lists, which can be easily implemented with arrays. Moreover, the extraction of a given level is linear with respect to the number of triangles at that level.

1. Introduction

A variety of applications use polygonal meshes for efficient visualization of complex surfaces. Technological advances favour the creation of meshes composed of thousand of polygons; for example, satellite data acquisition for terrain representation or laser scanning systems used in several areas, provide models of high complexity. The resulting meshes are very complex, and therefore expensive to store, transmit and render.

The use of simplification methods [1][2][3] is a solution to this problem that achieves a compact approximation of the original model with a smaller set of polygons and guaranteed approximation error, reducing disk and memory requirements and speeding network transmission.

To further improve rendering performance, it is common to define various levels of detail (LOD) of the model [4]. The coarsest approximations are used when the viewer is far from the object, while higher detail versions are substituted as the viewer approaches. However, visual discontinuities are encountered when switching between two approximations. Multiresolution modeling [5] tries to solve this problem by maintaining multiple levels of detail.

Lounsbery et al. [6] generalize the concept of multiresolution analysis (MRA) for surfaces of arbitrary topological type. Eck et al. [7] present a method for overcoming the subdivision connectivity restriction, meaning that completely arbitrary meshes can be converted to multiresolution form. Certain et al. [8] extend MRA to perform multiresolution analysis of colored meshes by separately analyzing shape and color.

Hoppe [9] presents the progressive mesh (PM) representation, a new scheme for storing and transmitting arbitrary triangles meshes. It supports geomorphs, progressive transmission, compression and selective refinement. In PM form, an arbitrary mesh M is stored as a much coarser mesh M^0 together with a sequence of n detail records that indicate how to incrementally refine M^0 back into the original mesh M. Xia and Varshney [10] extend PM, with the aim of performing real-time adaptive simplifications and refinements. A merge tree is constructed over the vertices of mesh M so that one can reach any child vertex in O($log\ n$) links.

Ciampalini et al. [11], present a general multiresolution data scheme. All triangles generated during the simplification process have two time stamps: the birth and death time. An interval tree is used to retrieve all the facets whose life intervals cover a given precision. The complexity of retrieval is O($m\ log\ k$) where m is the number of triangles and k the number of decimation steps.

Puppo [12] presents a model based on a collection of fragments of plane triangulation arranged into a partially ordered set. An efficient algorithm, linear with respect to

the number of triangles at that level, is proposed. Klein et al. [13] improve the extraction of a LOD with a new algorithm that is able to refine a given triangulation in a certain area and to coarsen it in another area.

Other multiresolution models that use binary space partitioning techniques (BSP Trees) are still in development. They solve the problem only for 2½D[14] [15]. However, the point of view of BSP trees is very interesting given that a hidden surface algorithm is not required.

In order to construct the multiresolution model proposed here (MOM), a simplification process is necessary to generate the set of approximation meshes. Simplification 'on the fly' is difficult, expensive and inefficient. However, it can be run as a preprocess off-line, thus allowing interactive extraction of any LOD representation. Among the existing simplification algorithms, the method based on vertex decimation proposed by Schroeder et al. [16] is used in our experiments. However, MOM does not depend on the particular simplification approach adopted.

The aim of this work is to construct a multiresolution model, which satisfies the following criteria:

- Mesh extraction for a given resolution is done in real-time.
- The extracted mesh is continuous, that is, it does not contain any break.
- Transition between two meshes must be smooth, avoiding abrupt changes.
- The model is lossless.

We propose a data structure for storing the set of approximation meshes and an algorithm to extract a representation with continuous resolution, thus allowing a fast and interactive visualization of the model. We also describe the process to transmit the model progressively.

The rest of paper is organized as follows. In section 2 we present the data structure, the model construction, an analysis of the storage cost, the algorithm to extract a LOD, the process to transmit the model and a comparison between MOM and other multiresolution models. In section 3, we show the results of several experiments. Finally, section 4 contains conclusions and ideas for future work.

2. MOM: The Multiresolution Model

2.1. Data Structure

The data structure used in the multiresolution model consists of three sets of data:

- A list of vertices, with coordinates of all vertices used in the representation of any LOD. They are ordered depending on the simplification process: the first vertices to be deleted are the first vertices in the list. In addition, each vertex has a list of the triangles surrounding it when it is deleted. Triangles common to more than one of these deleted vertices appear just once. During the simplification process, the new vertices are added at the end of the list at consecutive positions.
- A list of triangles, with all triangles that can be used in the representation of any LOD. Only two indices (or pointers) define a triangle, instead of three as will be described later. The list is initialized with all the triangles of the original mesh. During the simplification process, the new triangles are added at the end of the list at consecutive positions.
- A control list, that stores for each LOD how many vertices do not belong to it and how many triangles are candidates for representing it.

2.2. Model Construction

To explain the construction process we use a mesh with nine vertices and nine triangles (figure 1). This is the original mesh. Two new LOD will be stored to complete the multiresolution model. For simplicity, let us assume that only one vertex is deleted in each simplification step, and the generated hole is retriangulated [16].

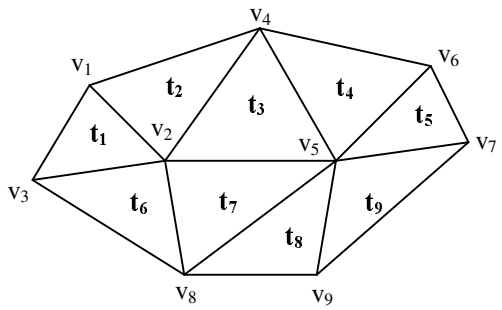

Figure 1. The original mesh.

We start from a data structure with a list of vertices and a list of triangles. Vertices are stored just once in the list and triangles are defined by three indices (or pointers) to the list of vertices [17]. In addition, a control list is necessary to indicate that there are no deleted vertices and that all triangles will be displayed. This representation stores the higher level of detail (figure 2). For storing a new level of detail we need:

- The vertices to be deleted, for example, v_5.
- The triangles to be deleted t_3, t_4, t_5, t_7, t_8, *and* t_9.
- The new triangles generated t_{10}, t_{11}, t_{12}, *and* t_{13}.
- The new vertices generated, none in this case.

The new mesh (figure 3) is stored in the data structure by the following five steps:

1. Put v_5 in the correct ordered position in the list, that is, after the last deleted vertex. In this case, as it is the first deleted vertex, it will be the first vertex on the list.
2. Associate to v_5 all deleted triangles t_3, t_4, t_5, t_7, t_8, *and* t_9.
3. Add to the list of triangles the new triangles t_{10}, t_{11}, t_{12}, *and* t_{13}.
4. Add to the list of vertices the new vertices, if any.
5. Add to the control list a new LOD, storing how many vertices have been deleted and how many triangles are candidates for representing it. In this case, 1 vertex and 13 triangles.

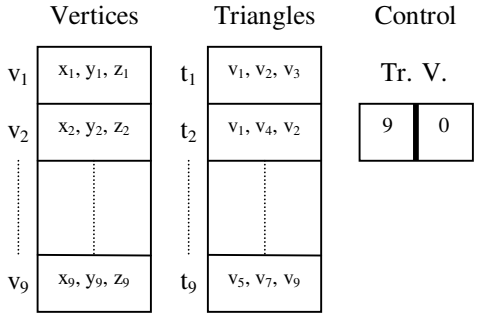

Figure 2. Representation of the original mesh.

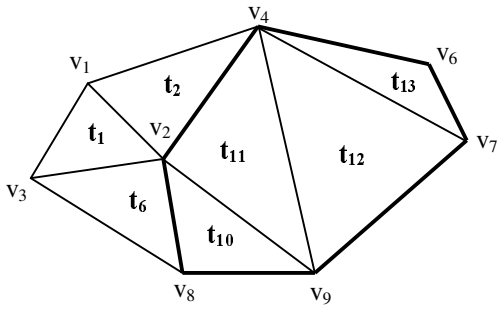

Figure 3. Mesh after v_5 simplification.

Figure 4 shows the updated data structure with the new LOD.

If the simplification process selects v_2 as the next vertex to be deleted and proceeding in the same way, we would obtain the mesh shown in figure 5. Again, the updated data structure with this new LOD is shown in figure 6.

Note that the list of triangles associated to v_2 contains triangles that belong to different levels of detail. In this case, they must be ordered in the same order of appearance in the list of triangles. This is necessary to increase the visualization algorithm efficiency.

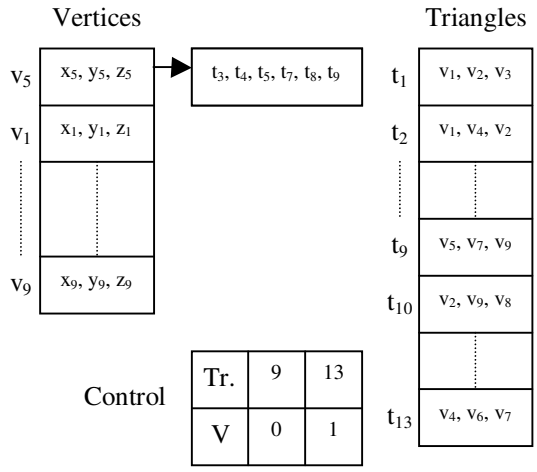

Figure 4. The model with two LOD.

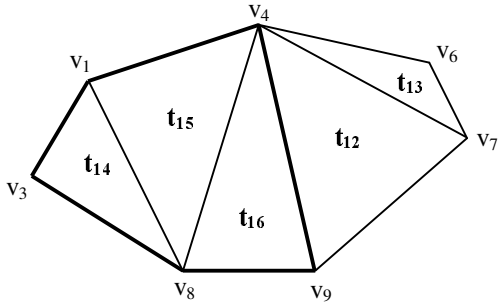

Figure 5. Mesh after v_2 simplification.

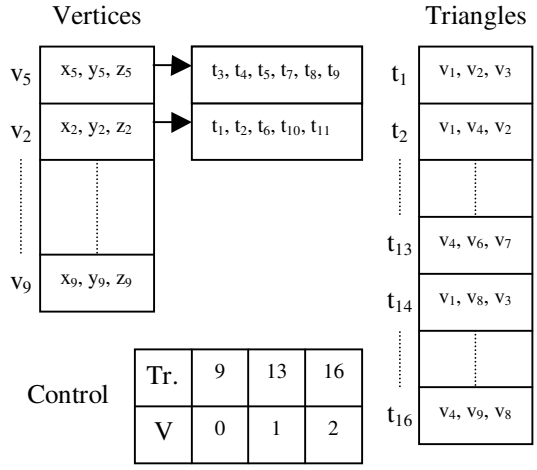

Figure 6. The model with three LOD.

After finishing the simplification process, a last step must be done:

- Distribute the triangles, which belong to the last LOD (t_{12}, t_{13}, t_{14}, t_{15}, t_{16}), among the non-deleted vertices (v_1, v_3, v_4, v_6, v_7, v_8, v_9).

Still, it is possible to further simplify this data structure. Given that vertices point to triangles and any

triangle is pointed by one vertex, at each triangle we store just the other two indices (or pointers) to the list of vertices. Figure 7 shows the resulting multiresolution model.

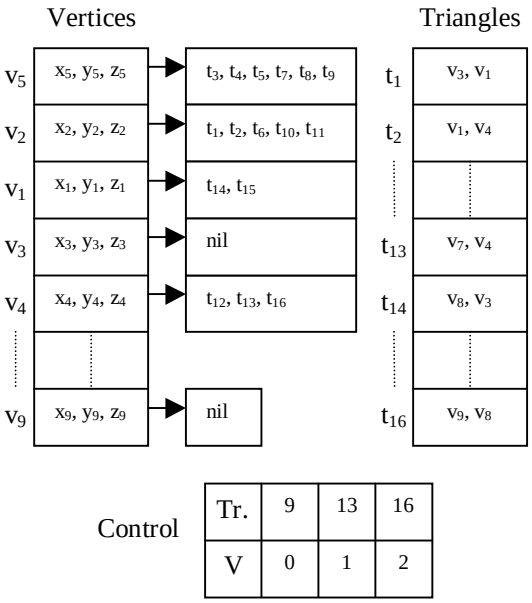

Figure 7. The Multiresolution Model.

As figure 7 shows, triangle t_1 (v_1, v_2, v_3) is associated to vertex v_2, so that it stores only v_3 and v_1.

2.3. Storage Cost

The space requirements of the data structure can be evaluated as follows. Let V and T be the number of vertices and triangles of the original mesh, V' and T' be the number of vertices and triangles generated during the whole simplification process and N be the number of levels of detail.

- List of vertices. Each vertex stores its coordinates (x, y, z) and one index (or pointer) to the first triangle of the associated triangle list. The original mesh is composed of V vertices. During the simplification process V' vertices were generated. The bound for V' depends on the simplification method, i.e. the method by Schroeder et al. [16] generates no new vertices. Thus, the total size is (V+ V')* (3 real* + 1 integer).
- List of triangles. The original mesh is composed of T triangles. During the simplification process T' triangles were generated. In general, there is not a linear bound for T'. In addition, each triangle stores just two indices (or pointers) to the list of vertices. Thus, the total size is (T +T') * 2 integer.
- List of triangles associated to the vertices. Each triangle is pointed to by just one vertex. Thus, the total size is T+ T' integer. Note that this storage cost is compensated by the space saved in the list of triangles.
- Control list. Depends on the number of levels of detail. Thus, the total size is N* 2 integer.

In table 1, the storage costs of monoresolution and multiresolution models are shown.

	Monoresolution	Multiresolution
List of Vertices	3V real	3 (V + V') real + (V + V') integer
List of Triangles	3T integer	2 (T + T') integer
List of Triangles associated to Vertices	None	T + T' integer
List of Control	None	2N integer

Table 1. Relationship between storage costs.

In section 3, results of some experimental models are shown.

2.4. LOD Visualization

The algorithm to extract a given LOD, that is, the set of triangles used to represent it, is shown in figure 8. Due to the ordering of vertices and triangles (order generated in the construction process) and the control information, the extraction cost is linear with respect to the number of triangles pertaining to that LOD.

```
Given a LOD, from control list extract:
i ← the first vertex belong to that LOD
t ← #triangles candidates for representing it

We calculate m as a function of i and t:
m ← #triangles pertaining to that LOD

/* while there be triangles to display */
while (m> 0)
    /* traverse the list of triangles associated to the vertex i*/
    while (not empty triangle_list(i)) and (triangle index < t)
        display triangle
        m= m - 1
    end while
    i = i + 1 /* proceed to the next vertex */
end while
```

Figure 8. Algorithm for extracting a given LOD.

The example in section 2.2 stores three levels of detail. Table 2 shows a summary of the triangles displayed for each LOD and the vertices that point to them.

* We denote by integer and real the amount of memory used for storing either an integer or a pointer, and a real number, respectively.

1st LOD		2nd LOD		3 rd LOD	
V=0, T= 9		V=1, T= 13		V=2, T= 16	
v_5	$t_3, t_4, t_5, t_7, t_8, t_9$	v_2	$t_1, t_2, t_6, t_{10}, t_{11}$	v_1	t_{14}, t_{15}
v_2	t_1, t_2, t_6	v_4	t_{12}, t_{13}	v_4	t_{12}, t_{13}, t_{16}

Table 2. Vertices and triangles represent each LOD.

In section 3, results regarding some experimental models are shown.

2.5. Progressive transmission

For progressive transmission, the coarsest mesh is transmitted first (using a conventional monoresolution format), followed by a set of records. Each record includes the information required for refining the mesh. This information depends on the simplification method. The receiving process rebuilds the original mesh, as the records arrive, and displays the successive updates.

In case of a closed mesh, if we use the method based on vertex decimation proposed by Schroeder et al. [16], a trio (v, n, t) can denote each record, where:

- v is the new vertex (x, y, z).
- n is the length of t.
- t is the set of triangles (only indices) affected by the new vertex v.

For example, the model in figure 7 would be transmitted following the sequence:

- Transmit the coarsest mesh: vertices ($v_1, v_3, v_4, v_6, v_7, v_8, v_9$) and triangles ($t_{12}, t_{13}, t_{14}, t_{15}, t_{16}$).
- Transmit the record ($v_2, 3, (14, 15, 16)$) to refine the triangles ($t_1, t_2, t_6, t_{10}, t_{11}$).
- Transmit the record ($v_5, 4, (10, 11, 12, 13)$) to refine the triangles ($t_3, t_4, t_5, t_7, t_8, t_9$).

Since, in general, there is not a linear bound for the number of triangles generated during the simplification process, it is not possible to set a bound on the size of the data to be transmitted. Again, in section 3, results regarding some experimental models are shown.

2.6. A comparison between MOM and other multiresolution models

In this section, we present a theoretical comparison between MOM and other existing multiresolution models, denoting the main differences.

Multiresolution Analysis, MRA [6]*:* following Hoppe's evaluations [9], our solution is a lossless representation and the original mesh can be recovered exactly. In addition, MOM is not restricted to meshes with subdivision connectivity. Therefore, the intermediate meshes can be better approximations than their MRA counterparts, and can manage with higher precision the feature edges or discontinuities of the mesh. However, MRA makes possible multiresolution surface editing and allows geometry and color to be compressed independently.

Progressive Meshes, PM [9]: this model is concise in terms of storage, but requires more computational effort to extract a LOD because it stores an ordered sequence of updates, where intermediate meshes are stored implicitly. The total number of child links to traverse before reaching the desired node is O(*n*).

Merge Tree, MT [10]: this model constructs a merge tree over the vertices of mesh M so that one can reach any child vertex in O(*log n*), where n is a sequence of detail records.

JADE [11]: the storage cost is higher than MOM because it stores two time stamps per triangle, and it uses two lists of triangles per node to extract the LOD efficiently [18]. The extraction cost, O(*m log k*), depends on the number of decimation steps, *k*, and the number of triangles, *m*. In MOM, the extraction cost only depends on the number of triangles *m*.

Multiresolution Model, MRM [12]: the storage cost of this model is more expensive. It stores a set of fragments for representing the whole multitriangulation. A fragment represents a remaining hole in the simplification process and contains indices of the bottom and ceiling triangles. In addition, each triangle stores two pointers to the fragment containing it. The progressive transmission is possible [13] but it needs to transmit all the fragments. MOM only stores the indices of the ceiling triangles and only transmits the indices of the bottom triangles.

We would like to note that an important feature of MOM is its simplicity. Instead of a tree structure, a list-based structure is proposed, which can be easily implemented with arrays.

3. Results

Time measurements are obtained on a Pentium II based workstation (266Mhz, 64Mb of SDRAM) and a graphics accelerator for OpenGL based on Permedia 2. The simplification method used in the experiments is based on vertex decimation, proposed by Schroeder et al. [16].

Figure 9 shows four different LOD's of the bunny model, composed of 69473 faces (triangles) and 34835 vertices. Table 3 shows the extraction and visualization times for each LOD.

Figure 10 shows four different LOD's of the sphere model, composed of 2208 faces (triangles) and 1106

vertices. Table 4 shows the extraction and visualization times for each LOD.

From table 3 and 4, we can observe that the extraction time for the monoresolution models is slightly lower than extracting the equivalent LOD in MOM. However, using a lower LOD without losing the required visual quality reduces considerably the extraction time.

BUNNY	Mono.	LOD A	LOD B	LOD C	LOD D
Vertices	34835	34835	7267	3548	778
Triangles	69474	69474	14338	6900	1360
Deleted Vertices	-	0	27548	31287	33987
Extraction Time	34.08	45.67	8.41	3.81	1.01
Extraction + Visualization Time	200.00	235.30	48.09	23.00	5.00

Table 3. Relationship between extraction times (ms.).

SPHERE	Mono.	LOD A	LOD B	LOD C	LOD D
Vertices	1106	1106	553	225	102
Triangles	2208	2208	1102	466	200
Deleted Vertices	-	0	553	881	1004
Extraction Time	0.650	0.687	0.362	0.150	0.068
Extraction + Visualization Time	7.20	7.30	5.91	3.91	2.60

Table 4. Relationship between extraction times (ms.).

The storage costs of both examples are shown in table 5. The number of levels of detail is equal to the number of deleted vertices plus one (because no vertex is deleted in the first LOD), i.e. 33988 for the bunny model and 1005 for the sphere model. If there were no need to keep all the levels available, the total cost of the model would be reduced slightly.

	Monoresolution	Multiresolution	Progressive Transmission
Bunny	102k real + 204k integer	102k real + 685k integer	2.28k real + 4k int. + 99.72k real + 159k int.
Sphere	3.24k real + 6.47k integer	3.24 real + 18.57k integer	0.3k real + 0.6k int. + 2.94k real + 3.80k int.

Table 5. Storage cost.

The results presented in table 5 show that the storage cost of the multiresolution model is higher, between two and three times, than the monoresolution model. However, for the bunny model, MOM includes almost 34000 different LOD's. In addition, the compressed model for progressive transmission requires less space than both models.

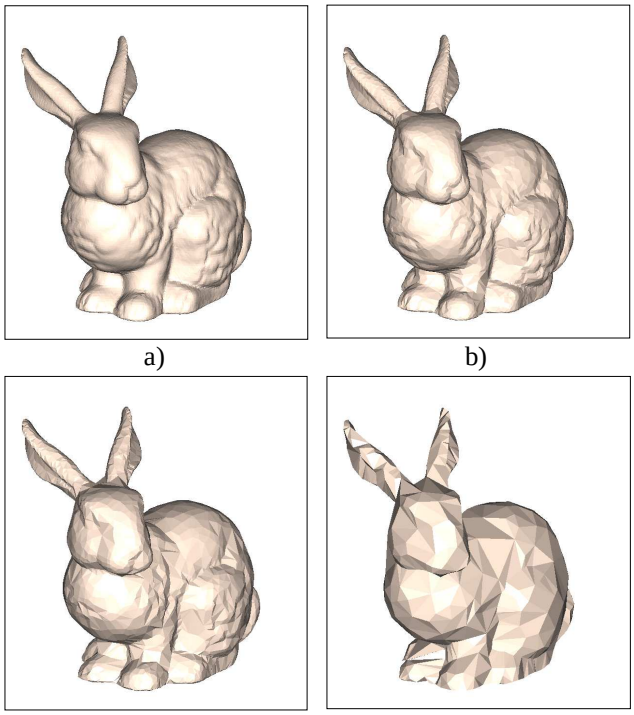

Figure 9. a) LOD A, b) LOD B, c) LOD C, d) LOD D.

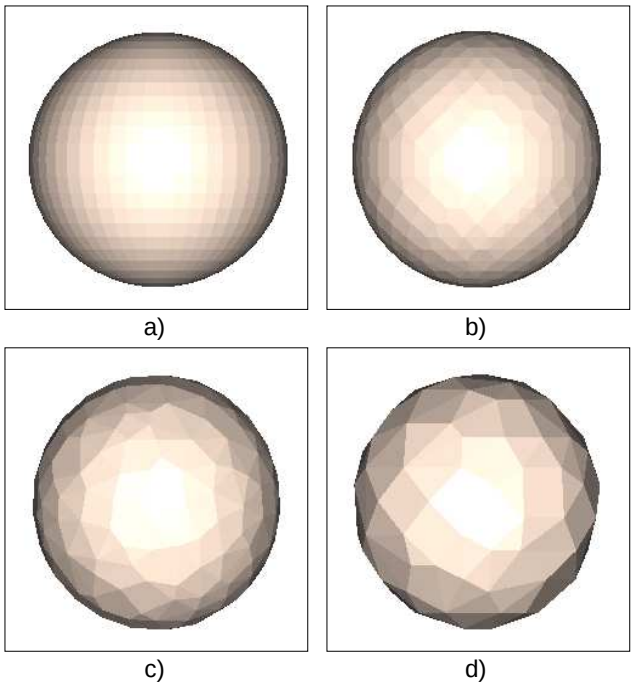

Figure 10. a) LOD A, b) LOD B, c) LOD C, d) LOD D.

4. Conclusions and Future Work

A new multiresolution model for representing complex triangles meshes with *n* levels of detail in a compact and simple way has been presented. Meshes are stored in order depending on the simplification process. Given a LOD, the extraction cost is linear with respect to the number of triangles pertaining to that LOD, so that it can be displayed in real time. Moreover, the multiresolution model does not depend on the particular simplification approach adopted.

The experimental results show that the multiresolution model requires more space than the monoresolution model, but the possibility of compressing the mesh produces a model with even less space than its better representation.

The main task still to be addressed is the selective refinement, or dependent view. It consists of displaying a model with higher detail in the required part, while using lower details in other parts of the mesh. The area with higher detail may be selected by either the user, or other factors for example, the view, the illumination, etc.

The structure does not store data like normal or color, which would improve considerably the quality of visualization. Future work will be directed to the inclusion of these data in the model.

5. Acknowledgments

This work was partially supported by grants P1A96-22 (Fundació Caixa Castelló-Bancaixa) and GV97-TI-05-42 (Generalitat Valenciana).

6. References

[1] P. Heckbert, M. Garland, "Survey of surface simplification algorithms", Technical Report, Carnegie Mellon University - Dept. of Computer Science, 1997.

[2] Enrico Puppo, Roberto Scopigno, "Simplification, LOD and Multiresolution Principles and Applications", EUROGRAPHICS'97, 1997, vol. 16, no. 3.

[3] C. Erikson, "Polygonal Simplification: An Overview", Technical Report TR96-016, Department of Computer Science, UNC-Chapel Hill, 1996.

[4] Thomas A. Funkhouser, Carlo H. Sequin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments", Computer Graphics (SIGGRAPH '93 Proceedings), 1993, Aug, vol. 27, pages 247-254

[5] Paul Heckbert, Michael Garland, "Multiresolution Modeling for Fast Rendering", Proceedings of Graphics Interface '94, Banff Alberta Canada, 1994, may, pages 43-50

[6] Michael Lounsbery, Tony DeRose, Joe Warren, "Multiresolution analysis for surfaces of arbitrary topological type", ACM Transactions on Graphics, 1994, vol. 16, no. 1, pages 34-73.

[7] Matthias Eck , Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, Werner Stuetzle, "Multiresolution Analysis of Arbitrary Meshes ", SIGGRAPH '95 Proc., Ed. ACM, 1995, Aug, pages 173-182.

[8] Andrew Certain, Jovan Popovic, Tony DeRose, Tom Duchamp, David Salesin, Werner Stuetzle, "Interactive Multiresolution Surface Viewing", SIGGRAPH 96 Conference Proceedings, Ed. Addison Wesley, 1996, Aug, pages 91-98.

[9] Hugues Hoppe, "Progressive Meshes", SIGGRAPH '96 Proc., 1996, Aug, pages 99-108.

[10] J. Xia , A. Varshney, "Dynamic view-dependent simplification for polygonal models", IEEE Visualization'96 Proc., Ed. R. Yagel, G. Nielson, 1996, pages 327-334.

[11] Ciampalini, P. Cignoni, C. Montani, R. Scopigno, "Multiresolution decimation based on global error", Technical Report C96-021, CNUCE-C.N.R. (To appear on The Visual Computer, 1997), Pisa, Italy, 1996, Jul.

[12] E. Puppo, "Variable resolution triangulation", Technical Report. 12/96. Institute for Applied Mathematics, C.N.R., Genova (Italy), 1996, Nov.

[13] Reinhard Klein, Jörg Krämer, "Building multiresolution models for fast interactive visualization", SCCG'97, 1997.

[14] J. Huerta, M. Chover, R. Quirós, R. Vivó, J. Ribelles, "Binary Spaces Partitioning Trees: A Multiresolution Approach", Proceedings Information Visualization, London, 1997, August, pages 148-154.

[15] J. Huerta, M. Chover, J. Ribelles, R. Quirós , "Construction and Rendering of Multiresolution Binary Space Partitioning Trees", Compugraphics'97, Algarve, Portugal, 1997, pages 212-221.

[16] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen, "Decimation of triangle meshes", Computer Graphics (SIGGRAPH '92 Proc.), 1992, Jul, vol. 26, no. 2, pages 65-70.

[17] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Richard L. Philips, "Introduction to Computer Graphics", Reading MA, Ed. Addison-Wesley, 1994.

[18] Paolo Cignoni, Paola Marino, Claudio Montani, Enrico Puppo, Roberto Scopigno, "Speeding Up Isosurface Extraction using Interval Trees", IEEE Transactions on Visualization and Computer Graphics, 1997, Mars, vol. 3, no. 4, pages 352-369.

[19] Julie C. Xia , Jihad El-Sana, Amitabh Varshney, "Adaptive Real-Time Level-of-detail-based Rendering for Polygonal Models", IEEE Transactions on Visualization and Computer Graphics, 1997, Jun.