MULTIRESOLUTION MODELLING USING CONNECTIVITY INFORMATION

Ó. Belmonte‡, I. Remolar‡, J. Ribelles‡, M. Chover‡, C. Rebollo‡, M. Fernández¶1

‡Departamento de Lenguajes y Sistemas Informáticos. Universitat Jaume I. Castellón. Spain.
Universitat Jaume I, Campus de Riu Sec.
12080, Castellón de la Plana, Spain
{belfern, remolar, ribelles, chover}@uji.es

¶Departamento de Informática. Universitat de Valencia.
46100, Valencia, Spain
marcos@robotica.uv.es

ABSTRACT

Triangles meshes are the most popular standard model to represent polygonal surfaces in Computer Graphics. Drawing these meshes as a set of independent triangles involves sending a vast amount of information sent to the graphic engine. The use of primitives such as fan and strip of triangles, which make use of the connectivity information between the triangles in the mesh, reduces dramatically the amount of information sent to the graphic engine. The Multiresolution Triangle Strips scheme takes advantage of this characteristic in order to represent a multiresolution model as a set of multiresolution triangle strips. A multiresolution triangle strip is made of the original strips and all of its Levels of Detail. Each of these multiresolution strips is represented as a graph that is traversed to recover the demanded LoD. A Multiresolution Triangle Strip model uses the triangle strip primitive both in the data structure as in the rendering stage. The Multiresolution Triangle Strip is compared against the Progressive Meshes multiresolution model, one of the best multiresolution models probably known. The performance of the MTS models in visualising improves drastically PM models.

Keywords : Multiresolution, triangle strip, real time rendering, computer graphics.

1. INTRODUCTION

Triangle meshes have become the standard model to represent polygonal surfaces. In several Computer Graphics applications as real-time visualisation, virtual reality, computer games, etc. surfaces are described by very dense triangle meshes. There are two main reasons: the simplicity of the drawing algorithm, easily implemented in hardware, and the fact that any mesh can be triangulated.

Nowadays, highly detailed geometric models with hundreds of thousands triangles are managed in many Computer Graphics applications. These models are very expensive to visualise. In some cases a simplified version, with a lower number of triangles, retains the visual appearance of the original model. For example, it is not necessary to represent an object that it is far from the viewer with a ten thousand triangles when it only covers one hundred pixels in the screen. The simplified version is said to have a lower *Level of Detail* (LoD).

Multiresolution models support the representation and processing of geometric entities at different levels of detail, depending on specific needs of the application. The common criteria to determine the most suitable level of detail are the distance of the object to the viewer, the projection area, the eccentricity of the object on the screen and the intrinsic importance of the object.

Current graphic systems are able to render more triangles than they actually receive. Nowadays, the bottleneck in the render stage is the throughput of the graphic systems in receiving the information to visualise. This amount of information decreases considerably if the connectivity property between the triangles is used in the mesh representation. Some graphic primitives as fans and strips of triangles take advantage of this property. These appear

[1]This work is partially supported by the Ministerio Español de Educación y Ciencia, grants TIC 1999-0510-C02-02

as drawing primitives in some graphics libraries, such as *OpenGL*.

All the multiresolution models in the literature, except *MOM-FAN* [Ribel2000] use the triangle primitive as base in both the data structure and in the rendering stage. Multiresolution Triangle Strip (MTS) is the first scheme that represents a multiresolution model using the triangle strip primitive in these two stages. A MTS model is composed by a set of multiresolution strips. A multiresolution strip represents the original strip and all of its LoDs. Each of these strips is represented as a directed graph with weights in its arcs.

In the following, the previous work is presented in §2 including search triangle strips algorithms in triangles meshes and multiresolution models that use the fan or strip of triangles primitive during the render stage. In §3 the MTS model is presented, its data structure and the recovery algorithm. We show with an example how a graph is constructed and how to recover a specific level of detail. In §4 the results are shown, comparing them to the results from Progressive Meshes [Hoppe1996]. Finally, in §5 conclusions and future work are presented.

2. PREVIOUS WORK

In this section, some works relative to the multiresolution model presented in this paper is briefly reviewed . First of all, searching strips algorithms over a polygonal surface are discussed. After this, we review the simplification algorithm by Garland and Heckbert [Garla1997a], that has been used to obtain the coarse meshes of the original model. Other multiresolution models that use fan or strip of triangles either in the data structures or in the render stage are discussed.

2.1 Triangle strip search algorithms

A strip of triangles uses the connectivity information to represent a polygonal surface. In fig. 1.a) an example of triangle strip is shown. This strip is codified as the vertices sequence 0,1,2,3,4,5,6,7, where the triangle *i* is composed by the vertices *i*, *i+1* and *i+2*. In this way, it is only necessary to send to the graphic system *T+2* vertices for rendering *T* triangles, against *3T* vertices that are necessary to send when the surface is rendered as a set of independent triangles. In some vertices sequences, a special operation, called *swap* is needed. Figure 1.b shows an example of this operation. The vertices sequence that represents the strip is 0,1,2,3,4,5,6,swap,7,8. As we can see, it is necessary to exchange the vertices 5 and 6 to represent the strip correctly. This operation can be simulated repeating some vertices, in this case the vertices sequence would be 0,1,2,3,4,5,6,5,7,8. A triangle strip that can be represented without any swap operation is called *sequential strip*; and a triangle strip that needs this operation, is called *generalised strip*.

Searching the best set of triangle strips in a mesh is an NP-complete problem [Arkin1996]. Thus, it is necessary to use some heuristic strategies in the search. The algorithm by Evans et al. [Evans1996] is used in the MTS model construction. This algorithm, called STRIPE, is in the public domain at http://www.cs.sunysb.edu/~stripe/. The STRIPE algorithm permits to control some parameters in the searching process, as the use of swaps operations.

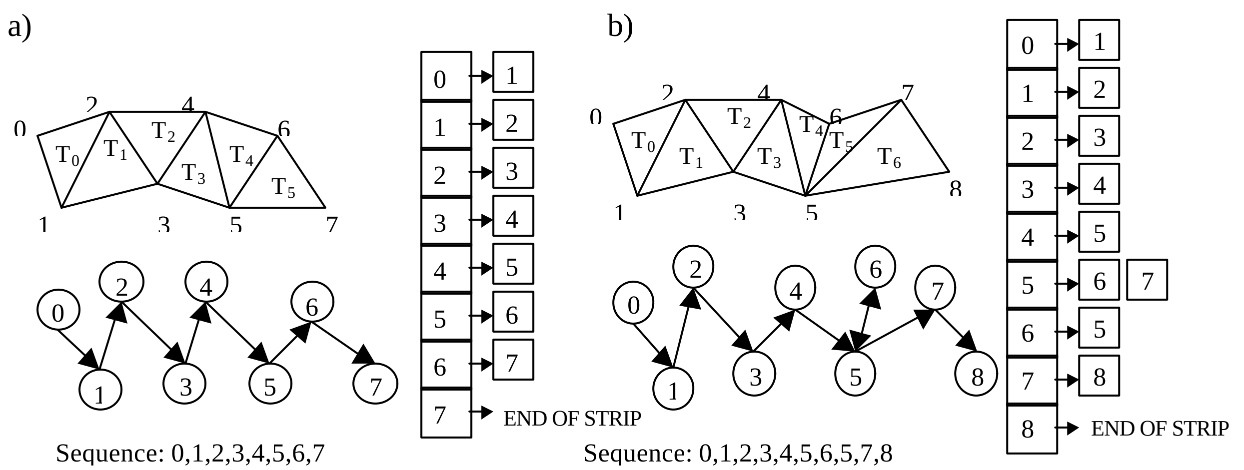

Figure 1: a) Example of strip, the vertices sequence that represents this strip and its associated graph.
b) Example of a strip with a swap operation and the graph that represents it.

2.2 Simplification using pair vertex contraction.

Every multiresolution model needs a simplification method that provides various geometric descriptions of the original polygonal surface with fewer geometric primitives. A simplified object has to maintain the appearance with the original model as much as possible. Similarity measure between an original mesh and a simplified one can be done using an appearance-based metric [Linds2000] or a geometric measure[Cigno1998].

There are important surveys where several simplification methods appeared in the literature [Garla1997b] are classified. Simplification algorithms based on iterative contraction are of particular interest because they have been used to construct multiresolution models representations.

The simplification method used in Multiresolution Triangle Strips is the method proposed by Garland and Heckbert [Garla1997a]. This method, called *Qslim*, is in the public domain and at http://www.cs.cmu.edu/~garland/quadrics/. It is based on iterative vertex-pair contraction. A 4x4 symmetric matrix $\mathbf{Q}_i$ is associated with each vertex $\mathbf{v}_i$. This matrix represents the distance from the vertex to the set of planes that share it. When a pair of vertices is contracted, their matrices are added together to form the matrix for the resulting vertex.

2.3 Multiresolution modelling.

Garland [Garla1999] defines a multiresolution model as a model representation that captures a wide range of approximations of an object and which can be used to reconstruct any one of them on demand.

The multiresolution models can be classified in two huge groups: *discrete multiresolution models*, that contain a discrete number of level of detail and a control mechanism to determine which is the most adequate one in every moment and the *continuous multiresolution models*, that capture a vast range of approximation of an object, virtually continuous. These can be subdivided into two main classes, according to their structure: tree-like models, and historical models [Puppo1999].

In a discrete multiresolution model there is no relation among the levels of detail of the object. Then, the size of these models increases rapidly when some new levels of detail are included. They usually store between five and ten levels of detail. Some graphic standards, as can be VRML or OpenInventor use discrete multiresolution models. These models are easily implemented and can be edited by the users and optimised for rendering. The main disadvantage is the *visual artefact* that occurs during the change between two levels of detail. A solution to decrease this visual artefact is to draw both levels of detail of the object using transparency methods. This solution increases the rendering time.

In continuous multiresolution models, two consecutive levels of detail differ in a few number of triangles. These little changes introduce a minimal visual artefact. On the other hand, the models size decreases in front of the discrete models because no duplicate information is stored. Progressive Meshes of Hugges Hoppe [Hoppe1996], is the most known continuous multiresolution model nowadays. It is included in the graphic library DirectX 8.0 from Microsoft.

2.3.1 Multiresolution models using triangle fans or strips primitives.

In this section, a review of the reduced number of models that uses the fan or strip of triangles primitive are revised. These models use these primitives either in the storage stage or the rendering stage.

Hoppe [Hoppe1997] has utilised strips in the rendering stage inside a view dependent multiresolution model. After selecting which triangles to draw, strips of triangles are searched. Through experimentation Hoppe concludes that the fastest triangle strip search algorithm is a greedy one. In this greedy algorithm, each of the non-drawn triangles begins a new strip, which grows through its non-rendered neighbours. In order to reduce the strip fragmentation, strips are grown in a spiral clockwise manner.

El-Sana et al. [El-San1999] have developed a view dependent multiresolution model based on an edge-collapsing criterion. The first step in constructing the model is to search triangle strips on it. These triangle strips are stored in a data structure called skip list [Pugh1990]. Once the multiresolution model has determined which triangles to visualise, the skip list is processed. If none of its triangles has been collapsed the strip is drawn. Otherwise the skip list is processed in order to update the strips. The triangle strips are not the basic primitive of the model, they are used to speed up the rendering process.

The work presented in [Ribel2000] modifies [Ribel1998] using fans of triangles as its basic representation primitive. Using this primitive, the storage cost is reduced, but the behaviour of this new model regarding its visualisation time is similar to its ancestor. A short average fan length, the high percentage of degenerate triangles, and the necessity to adjust the fans to the required LOD in real-time contribute to produce overall results which do

not suppose a global improvement in visualisation time.
Neither of the previous models use the strip of triangles primitive in both the storage and the rendering stage. Hoppe searches the strips over the simplified model previous rendering it. In the El-Sana work, we need to know which triangles to render for updating the original strips.

3 THE MTS MODEL

The main idea is to build the model as a set of multiresolution triangle strips. A multiresolution strip is made of the original strip and all of its level of detail. Each multiresolution strip is represented as a graph. The vertices sequence in a strip induces an order relationship between them. We conclude that the graph representing a multiresolution strip is a directed graph. Based on this basic structure the recovery level of detail algorithm is based on graph traversal.

In this section we describe data structures, how to build a multiresolution triangle strip and the level of detail recovery algorithm.

3.1 Data structures

A multiresolution strip is made of a graph, representing the strip in all levels of detail, and a list of strip beginnings (Listing 1: **class** MultiresolutionStrip).

Conceptually, each node in the graph represents a strip vertex and each directed arc of the graph represents an inner edge of the strip. The arc direction is determined by the vertices order in the sequence representing the strip. Two nodes joined by an arc in a graph are called adjacent. If the graph is a directed one an arc that joins a node v to another node w is incident from v and incident to w. In practice, a graph is represented by an adjacency list [Brass1996], see Fig. 2.c. In this representation, all nodes in the graph are elements of an array, and all incident nodes form another node belonging to its adjacency list.

Each strip vertex, represented by a node graph, has three fields. The first field is an index to the memory address where the geometric data of the vertex is allocated. The second field is a pointer to the adjacency list of the node. The third field is an index pointing to the next node which will be visited in the level of detail extraction process. See Listing 1, **class** ColumnNode.

Each inner edge of the strip, represented by an adjacency list element, has two fields. The first field is an index to the element in the column vector where the next vertex in the strip sequence is. The second field is an integer specifying the maximum level of detail at which the arc can be traversed in the level of detail extraction process. See Listing 1, **class** RowNode.In section 3.2 it will be shown the utility of this field with an example.
The list of strip beginnings specifies, for each level of detail, which is the initial vertex of the strip. The initial vertex would change as the strip is simplified. Each strip beginnings element has three fields. The first field specifies the index to the element in the column vector that is the beginning of the strip. The second field specifies the maximum level of detail to where the vertex is the beginning of the strip. This structure has the same fields as the elements in the adjacency list, class RowNode, so it has not been necessary to implement a specific class.

```
class ColummNode {
   unsigned long vIndex;
   unsigned int currentInd;
   RowNode * neighbours;
};
class RowNode {
   unsigned int colIndex;
   unsigned long res;
};
class MultiresolutionStrip {
   RowNode * sBegin;
   ColummNode * colVertices;
};
```

Listing 1: MTS data structures.

3.1.1 Construction examples

The construction of a multiresolution strips starts from the initial strip. As the strip is simplified by means of a pair vertex contraction, the vertex sequence changes. These changes in the vertex sequence should be introduced in the graph representing the strip.

Let's take as example the strip in Fig. 2.a). In this figure, the initial graph and the strip beginning list are shown. Let's label the maximum level of detail at which a strip can be represented as level of detail 0 (LoD 0). The vertices sequence at LoD 0 is 0,1,2,3,4,5,6,7,-3. The special label –3 specifies that the end of the strip has been reached. After the first pair vertex contraction the resulting vertices sequence is 1,2,3,4,5,6,7,-3. The beginning of the strip has become the vertex 1. This change is managed adding to the list of strip beginnings a new element, this element has 1 in its colndex field and has also 1 in its res field. After the second pair vertex contraction the new vertices sequence is
1,2,3,4,3,5,6,7,-3. At LoD 2 a swap operation between vertices 3 and 4 is needed. This change is managed adding two new elements, one two the adjacency list of vertex 4 (colIndex = 3, res = 2) and the other to the adjacency list of vertex 3 (colIndex = 5, res = 2). After the third pair vertex contraction

two strips appears having the common vertex 4. The special label –1 specifies that the 4 vertex is the end of one strips and also the beginning of the other one, so the vertices sequence at LoD 3 is 1,2,4,-1,6,7,-3. The new changes in the vertices sequence are managed adding the new elements as shown in Fig 2. After the last pair vertex contraction the new vertices sequence is 1,2,6,-3 at LoD 4.

3.2 Level of detail recovery algorithm

The level of detail recovery algorithm traverses the graph, which represents the multiresolution strip in order to extract the demanded level of detail. The algorithm proceeds in two steps. First, the algorithm finds out the vertex at the beginning of the strip, from the list of strip beginnings, with a resolution compatible with the demanded level of detail. Here compatible means that the element in the adjacency list has a field res that is bigger or equal to the demanded level of detail. Second, the graph is traversed from the vertex at the beginning until a special -3 node is reached. The pseudo code of the algorithm is shown in Listing 2.

```
// First we search the strip beginning
while BeginNotFound and NodesBeginning
  NextBeginning;
endwhile

if BeginNotFound exit //This strip does not
else                         //exist     at
this resolution
  while not EndStrip //While there are
                    //vertices in the strip
    while Neighbour.res < ResolutionDemmand
      nextNeighbour;
    endwhile

    if Node is not special node then
      DrawVertex;
    else if Node is -1 or Node is -2 then
      NewStrip;
    else if Node is -3 then
      EndStrip = true;
    endif
  endwhile
endif.
```

Listing 2: Recovery algorithm.

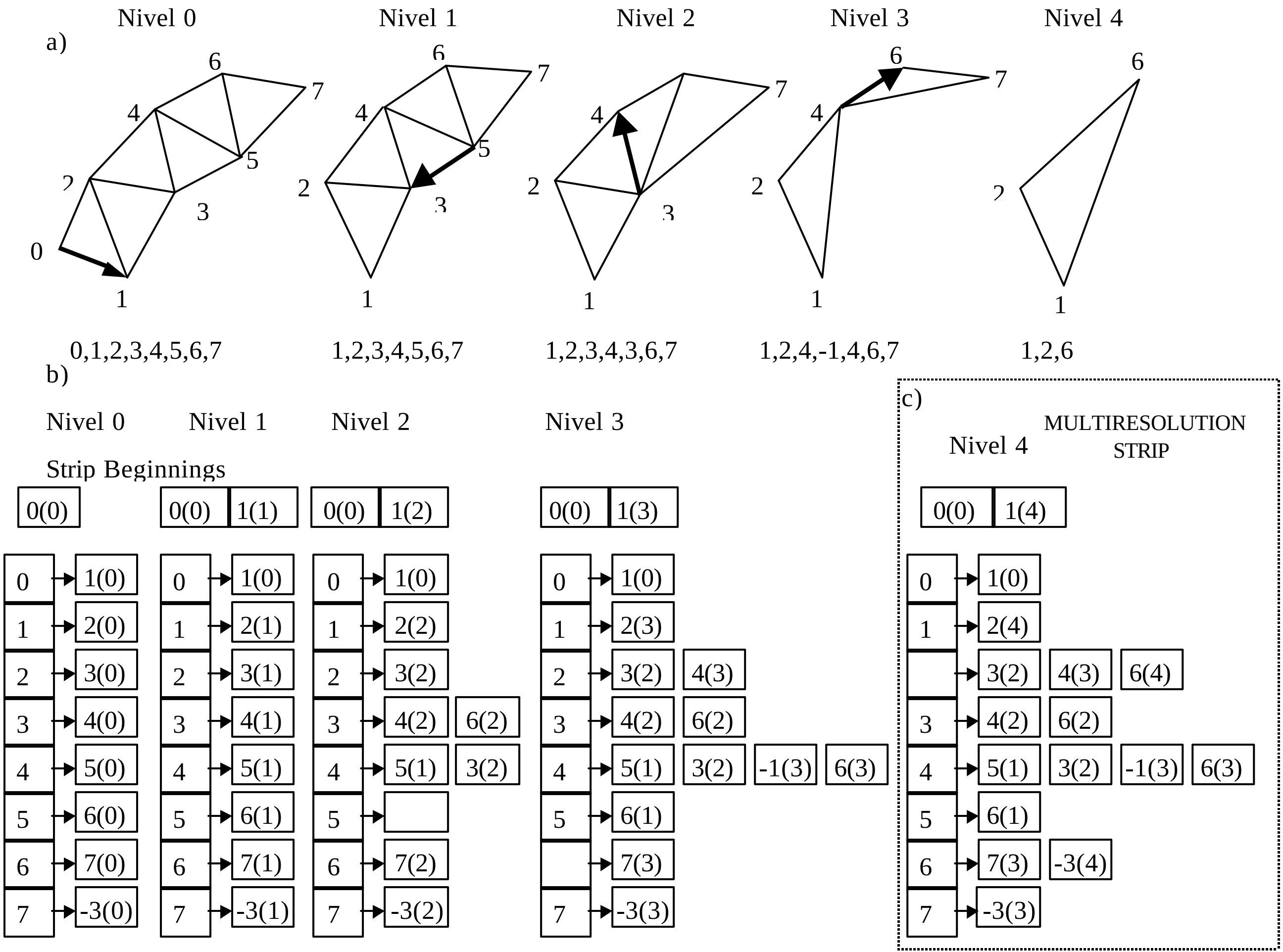

Figure 2: Construction of a multiresolution strip. a) Original and the sequence of pair vertex contraction. b) The detailed process of a multiresolution strip construction. c) The final multiresolution strip.

3.2.1 Recovery example

Given the multiresolution shown in Fig. 2. Let's suppose that the demanded LoD is 2. The first step of the algorithm is to find out the vertex at the beginning of the strip at this resolution. The search begins at the first element in the list of strip beginnings, and sequentially it searches the first element which res filed is bigger or equal to the demanded LoD. In this example the first element compatible is 1, so this is the beginning of the strip at LoD 2.

A previous step, before the extraction, is to update the field currentInd of each element in the column vector. In this way the search always begins at the first element in the adjacency list. The next step is to search through the list of adjacencies of the node at the beginning lf the strip that with a res field compatible with the demanded LoD. In this example the only node in the adjacency list has a res field compatible with the LoD demanded, so 2 is the next vertex in the sequence that represents the strip. After that, the currentInd field in the element at the column vector is updated in order to point to the next element in the list. The extraction continues until reach a vertex labelled –3, that, as we know indicates the end of the strip. Figure 3.a shows the nodes extracted to recover the LoD 2. Figure 3.b shows the nodes extracted to recover the LoD 4.

4 RESULTS

The MTS model has been subjected to several tests. These tests are addressed to evaluate the visualization time in a real time application. The results are compared with those of the Progressive Meshes (PM) model, one of the probably best-known models. PM uses the triangle primitive both in the data structures and in the rendering stage.

Here we use our own implementation of PM, this has been tested obtaining the same results that published by the author.

The polygonal objects used in the test come from the *Stanford University Computer Graphics Laboratory* (http://www-graphics.stanford.edu/data/3Dscanrep/) and *Cyberware* (http://www.ciberware.com/models/).

Model	#Strips	#Triangles	#Vertices
Cow	136	5804	2904
Bunny	1229	69451	34834
Sphere	173	30624	15314
Phone	1747	165963	83044

Table 1: Characteristics of the models.

The tests have been performed in an HP Kayak XU with two Pentium III processors at 550 MHz and 1 Gb. of main memory. A GALAXY by Evans & Sutherland with 15 Mb. video was used.

4.1 Constrution time and graph size

A previous step is the construction of the model. This task is done just one time for every object. In order to obtain the size of the model grpahs it is supposed that an integer is 2 bytes size and a pointer or long integer is 4 bytes size. Table 2 shows the construction time and the object size for the objects used in the tests.

Model	Time (MTS)	MTS (Mb.)	PM (Mb.)
Cow	31’’	0,286	0,256
Sphere	1 h. 36’ 44’’	2,145	1,318
Bunny	3 h. 19’ 6’’	5,301	2,993
Phone	37h 52’ 20’’	15,598	7,144

Table 2: Construction time for the MTS models and its size compair with those for PM.

a) Demanded LoD = 2 b) Demanded LoD = 4

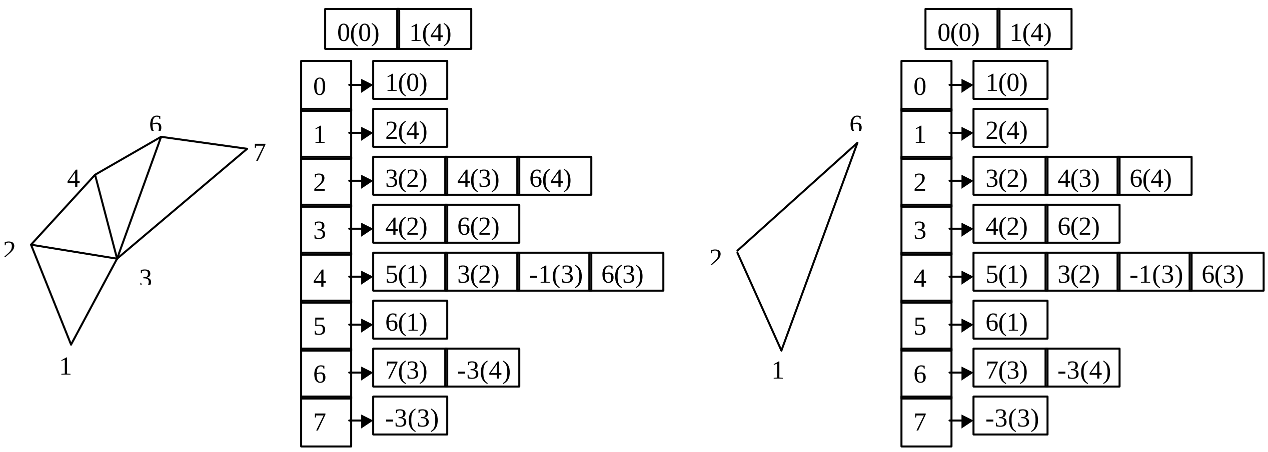

Figure 3: Two examples of the LoD recovery algorithm. a) The demanded resolution is 2. b) The demanded resolution is 4. The visited nodes are shown in grey back.

4.3 Visualization time

The distance between the object and the viewer has been used as criterion for select the demanded LoD. MTS models have been compared with PM models using this criterion. The great distance between the object and the viewer the great LoD demanded. Exponential and linear behaviours have been used in this criterion. The exponential behaviours produces big variations in the LoD when the object is closed to the viewer and small variations when the object is far from the viewer. This behaviour simulates the real world one.

Results are shown in Fig 4. Figures 4.a,c,e,g show the frame rate in a 'walkthrough' of each model with an exponential behaviour. Figures 4.b,d,f,h show the 'walkthrough' with a linear behaviour. The model moves away from the viewer as time goes by. The x-axis is the walkthrough time; the y-axis is the frame rate.

The bigger number of triangles in the model the lower frame rate is. MTS models frame rates are always above 10 frames per second. PM Phone model frame rate is 5 for LoDs closed to 0, where the model has more triangles. Starting from xx LoD the MT Phone model obtains better frame rates that the MTS model. This is because the saving in the visualization time in MTS gets lost in the recovery algorithm.

5 CONCLUSIONS AND FUTURE WORK

This paper presents a new multiresolution model. The main contribution is the use of the triangle strip primitive as basis in the data structure as in the rendering stage. The decrease in the amount of information sent to the graphic engine is the main advantage. This reduces the bottleneck towards the graphic engine speeding up the rendering stage.

MTS has been compared as against PM. The MTS model sizes are bigger than PM models. A task for future work is to reduce the size of the model taken advantage of the duplicated information stored in the graph that represents a multiresolution triangle strip.

The MTS visualization time is lower than PM. This can be extrapolated to all multiresolution models having the triangle primitive as basis in the data structure. A task for future work is to check this statement comparing MTS with other multiresolution models bases on the triangle strip primitive.

The main MTS disadvantage as against PM is the bigger recovery time to extract the demanded LoD. This is because PM uses coherence among two consecutive LoD extractions. A future work is to introduce coherence in MTS.

REFERENCES

[Arkin96].E. M. Arkin, M. Held J. S. B. Mitchell S. S. Kkiena, Hamiltonian Triangulations for fast Rendering, *Visual Computer, 12(9)*, 429-444, 1996.

[Puppo1999] E. Puppo, and R. Scopigno, Simplification, LOD and Multiresolution - Principles and Applications, *Tutorial Notes of EUROGRAPHICS'99*, 1999.

[Brass1996] G. Brassard and P. Bratley, Fundamentals of algoritmics, *Prentice Hall*, 1996.

[Hoppe1996] Huges Hoppe, Progressive Meshes, *Proceedings of SIGGRAPH '96*, 99-108, 1996.

[Hoppe1997] Huges Hoppe, View-dependent refinement of progressive meshes, *Proceedings of SIGGRAPH '97* , 189-197, 1997.

[El-San1999] J. El-Sana E. Azanli, and A. Varshney, Skip Strips: Maintaining Triangle Strips for View-dependent Rendering, . *IEEE Visualisation '99*, 131-138, 1999.

[Ribel2000] J. Ribelles, A. López I. Remolar O. Belmonte and M. Chover, Multiresolution Modelling of Polygonal Surface Meshes Using Triangle Fans. *Proceedings of 9th Discrete Geometry for Computer Imagery Conference* , 431-442, 2000.

[Ribel1998] J. Ribelles, M. Chover J. Huerta and R. Quirós, Multiresolution Ordered Meshes, *Proceedings of 1998 IEEE Conference on Information Visualization IV '98*, 198-204, 1998.

[Garla1999] M. Garland, Multiresolution Modelling: Survey & Future Opportunities, *State of the Art Reports of EUROGRAPHICS '99*, 111-131, 1999.

[Garla1997b] M. Garland, and P. Heckbert, Survey of polygonal surface simplification algorithms. *Course Notes of SIGGRAPH'97*, 1997.

[Garla1997a] M. Garland, and P. Heckbert, Surface Simplification Using Quadric Error Metrics, *Proceedings of SIGGRAPH '97*, 209-216, 1997.

[Cigno1998] P. Cignoni, C. Rocchini and R. Scopigno, Metro: measuring error on simplified surfaces, *Computer Graphics Forum*, 167-174, 1998.

[Pugh1990] W. Pugh, Skip lists: A probabilistic alternative to balanced trees, *Communications of the ACM, Vol 33(6)*, 668-678, 1990.

APPENDIX A

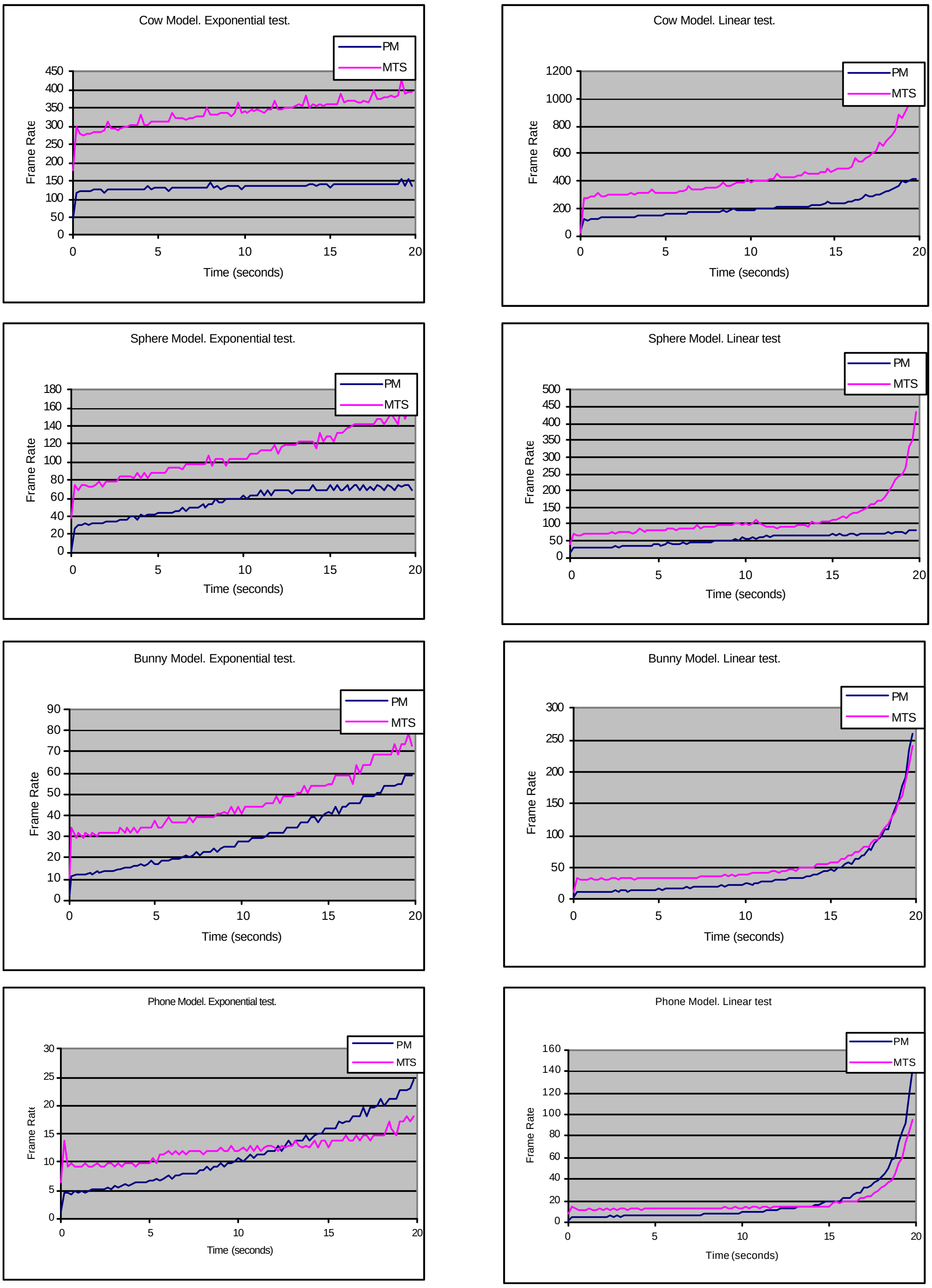

Figure 4: Experimental tests.