Available online at www.sciencedirect.com
SCIENCE DIRECT®
Future Generation Computer Systems 20 (2004) 1263–1273
FGCS FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/future

# Efficiently using connectivity information between triangles in a mesh for real-time rendering

Ó. Belmonte[a,*], I. Remolar[a], J. Ribelles[a], M. Chover[a], M. Fernández[b]

[a] *Department of Computer Languages and Systems, Universitat Jaume I, 12071 Castelló, Spain*
[b] *Department of Computer Science, Universitat de València, 46100 València, Spain*

Available online 7 July 2004

**Abstract**

Triangle meshes are the most popular standard model used to represent polygonal surfaces. Drawing these meshes as a set of independent triangles involves sending a vast amount of information to the graphics system. Taking advantage of the connectivity information between the triangles in a mesh dramatically diminishes the amount of information the graphics system must handle. *Multiresolution Triangle Strips* (*MTS*) represent a triangle mesh as a collection of *multiresolution triangles strips*. These strips are the basis of both the storage and the rendering stage. The coherence between the extraction of two levels of detail is used in the model in order to decrease the visualisation time.


*Keywords:* Multiresolution modelling; Level of detail; Interactive visualisation

## 1. Introduction

Triangle meshes have become the standard model with which to represent polygonal surfaces in Computer Graphics. The main reasons are the simplicity of the algorithms for drawing triangles, they are easily implemented in the hardware and the fact that any polygon with a number of sides greater than three can be broken down into a set of triangles.

Nowadays, highly detailed polygonal models are present in many Computer Graphics applications. These models contain a large number of triangles in their description. Rendering these models involves sending a vast amount of information to the graphics system.

It is not always necessary to render a fully detail polygonal model. In an early work [5], Clark points out that "It makes no sense to use 500 polygons in describing an object if it covers only 20 raster units of the display". In this case the model could be replaced by an approximation, adapting the number of triangles of a mesh to the needs of each application. This approximation is said to have a lower *Level of Detail* (LoD).

A multiresolution model supports the representation and processing of geometric entities at different levels of detail, depending on specific application needs. The common criteria to determine the most suitable LoD are [14]:

* Corresponding author.
*E-mail address:* belfern@uji.es (Ó. Belmonte).

0167-739X/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2004.05.028

- the distance of the object from the viewer;
- the projected area of the object on the display;
- the eccentricity of this object on the display;
- the intrinsic importance of the object.

Nowadays, current graphics systems can render more triangles than they actually receive. The bottleneck in the rendering stage is the throughput of the graphics systems in receiving the information to be visualised. Fortunately, this amount of information decreases considerably if the connectivity information between triangles in the mesh is used in their representation. Graphics primitives like fans and strips of triangles make use of this information. These graphics primitives appear in the majority of graphics libraries, such as *OpenGL* [18] and *IRIX-GL*. Modelling a mesh as a collection of fans or strips of triangles avoids the need to send a large amount of redundant information to the graphics system. All multiresolution models that have appeared in the literature, except *Multiresolution Ordered Meshes with Fan* (*MOM-Fan*) [17], take the triangle primitive as their base in both the storage and the rendering stage. Other multiresolution models like *View-Dependent Progressive Meshes* (*VDPM*) [12] and *Skip-Strips* [6] use the triangle strip primitive but only at the rendering stage.

*Multiresolution Triangle Strips* (*MTS*) is the first multiresolution model to represent an object using the triangle strip primitive in these two stages. An MTS model cosists of a collection of multiresolution strips and a multiresolution strip represents the original strip and all its LoDs.

In Section 2, the concept of triangle strip and the problem of searching for the best collection of triangle strips in a mesh are reviewed. The simplification method by Garland and Heckbert and multiresolution models that make use of fans or strips of triangles during the visualisation process are also considered in this section. In Section 3 MTS is presented, with special emphasis on its data structure. An illustrative example is used to show how a multiresolution strip is constructed. The model sizes of MTS are then compared with the sizes of *Progressive Meshes* (*PM*) [11] and MOM-Fan models. In Section 4 it is shown how coherence is incorporated into the model to reduce the extraction time of a particular LoD. In Section 5 time results of MTS are shown and compared with the results obtained for PM, MOM-Fan and Skip Strips. Finally, in Section 6 conclusions and future work are presented.

## 2. Previous work

In this section some previous work dealing with the multiresolution model presented in this paper are reviewed. The concept of triangle strips and an algorithm for searching for strips over a polygonal surface are discussed. The simplification method by Garland and Heckbert [8] is then reviewed. This simplification method was used to obtain the coarse meshes from the original model. Finally, other multiresolution models that use the fan or the strip of triangles, either at the storage or at the rendering stage, are discussed.

### 2.1. Strips of triangles

A strip of triangles encodes a sequence of triangles where every two sequential triangles share a common inner edge. Making use of this connectivity information decreases the amount of information sent to the graphics engine at the rendering stage. A triangle strip is encoded by a sequence of vertices. The sequence of vertices for the strip in Fig. 1 left is 0, 1, 2, 3, 4, 5, where the triangle $i$ is made up of vertices $i$, $i+1$ and $i+2$. In this way, only $T+2$ vertices need to bi sent to the graphics system to render $T$ triangles, while $3T$ vertices would be required if the triangles were rendered independently.

The sequence of vertices for the strip in Fig. 1 right, is 0, 1, 2, 3, 4, *swap*, 5, 6. In this case it is necessary to *swap* vertices 3 and 4 before rendering triangle T3. This swap operation can be emulated by sending vertex 3 twice. A triangle strip that can be represented without any swap operation is called a *sequential strip*. A triangle strip that needs this operation is called a *generalized strip*.

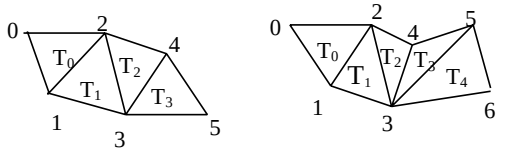


Fig. 1. Left, example of a strip and the sequence of vertices that represent this strip. Right, example of a strip with a swap operation and its sequence of vertices.

### 2.2. Searching for triangle strips

A triangle mesh can be drawn as a collection of triangle strips. From a real time rendering point of view, the best collection of triangle strips describing a mesh sends the lowest number of vertices to the graphics system. Searching for this best collection of triangle strips in a mesh is an NP complete problem [1]. Is, therefore necessary to use heuristic strategies to approach the problem in an affordable way.

A variety of triangle strip search algorithms have appeared in the literature. They can be classified into three main groups:

1. Local algorithms, wich search for the next triangle to be added to a strip within the vicinity of the last inserted triangle.
2. Global algorithms, which search for the next triangle in the whole model.
3. Hybrid algorithms, which first identify regions in the whole model and then use a local search over these regions.

A well-known example of a hybrid search algorithm is the STRIPE algorithm by Evan et al. [7]. This algorithm can be found at http://www.cs.sunysb.edu/stripe/. This algorithm works as follows: after identifying quadrilateral regions over the polygonal model, this algorithm uses a local search on them. If there is more than one candidate triangle to be added to the strip, the algorithm can choose the triangle that does not introduce a swap operation in the description of the strip. This was the option chosen to reduce the amount of information sent to the graphics system.

### 2.3. Simplification using pair vertex contraction

A simplification method applied to a triangle mesh yields a new mesh with fewer triangles than the original one but which maintains the same visual appearance. The difference in quality between the simplified triangle mesh and the original is evaluated using an error metric. This metric can be based on the geometry field [4] or on the image field [13].

Important surveys about simplification methods [8,9,15], have been published. The group of simplification methods based on vertex pair contraction has attracted the most interest in recent years because they can easily be used to construct multiresolution models.

The simplification method by Garland and Heckbert [8], called *Qslim*, is based on vertex pair contraction. In this method a $4 \times 4$ symmetric matrix is associated to each vertex. This matrix represents the distance between the vertex and the set of planes that share it. After a contraction, the sum of the two original matrices is assigned to the new vertex. The vertex pairs are contracted sequentially, and the first contraction involves the vertex pair that is closest to the set of planes that share it.

### 2.4. Multiresolution modelling

Garland [10] 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.

Multiresolution models can be classified into two large groups [14]: *discrete multiresolution models*, which contain a discrete number of LoDs and a control mechanism to determine which LoD is the most adequate in each moment; and *continuous multiresolution models*, which capture a vast range of approximations of an object in a virtually continuous manner. These can be subdivided into two main classes according to their structure: tree-like models, and historical models [15].

There is no relation between the LoDs of the object in a discrete multiresolution model. Thus, the size of these models increases quickly when some new levels of detail are included. They usually store between five and ten LoDs. Graphics standards such as VRML or OpenInventor use discrete multiresolution models. These models are easy to implement and can be edited by the users and optimised for rendering. The main disadvantage is the *visual artifact* that occurs during the change between two levels of detail. One solution to decrease this visual artifact is to draw both levels of detail of the object using transparency methods.

Two consecutive levels of detail differ differ by a small number of triangles in a continuous multiresolution model. These small changes introduce a minimal visual artifact. The model sizes decrease as compared to the discrete models because no duplicate information is stored. Nowadays, Progressive Meshes, by Hoppe [11], is the best known continuous multiresolution model. It is included in the graphics library DirectX 8.0 from Microsoft.

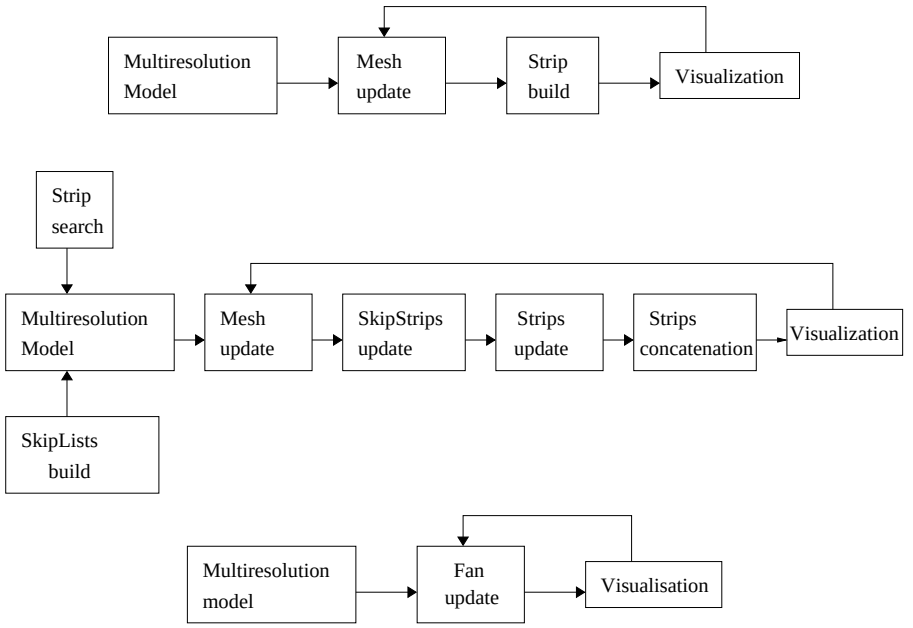


Fig. 2. Visualisation process of VDPM, Skip Strips and MOM-Fan models.

*2.4.1. Multiresolution models that use triangle fan or strip primitives*

There are a few multiresolution models that use a graphics primitive alternative to triangles at some stage during the visualisation process. Yet none of them use the triangle strip primitive both in the data structures of the model and in the rendering stage.

Hoppe [12] has utilised the triangle strip primitive in the rendering stage within a view-dependent multiresolution model. After selecting which triangles to draw, a collection of triangle strips is searched for over the mesh and, finally, these triangle strips are rendered. Through experimentation Hoppe concludes that the fastest triangle strip search algorithm is a greedy one. In order to reduce strip fragmentation, strips are grown in a clockwise spiral manner.

El-Sana et al. [6] have developed a view-dependent multiresolution model based on an edge collapsing criterion. The first step in constructing the model is to search for triangle strips on it. These triangle strips are stored in a data structure called a skip list. 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. Yet the triangle strips are not the basic primitive of the model, they are only used to speed up the rendering process.

The work presented in [17] uses fans of triangles as its basic representation primitive. Using this primitive, the storage cost is reduced but the behaviour of this new model as regards its visualisation time, is similar to its predecessor. A short average fan length, the high percentage of degenerate triangles, and the need to adjust the fans to the required LoD in real time contribute to produce results which do not give rise to an overall improvement in visualisation time.

Neither of the previous models uses the strip of triangles primitive in both the storage and the rendering stage. Hoppe searches for the strips over the simplified model prior to rendering it. In El-Sana's work, we need to know which triangles to render in order to update the original strips. Fig. 2 shows the visualisation process of the models described above.

## 3. The MTS model

An MTS model represents a mesh as a collection of multiresolution triangle strips. Each multiresolution triangle strip has all the necessary information to re-

cover the triangle strip at the required resolution. The steps needed to build up an MTS model are:

(1) to search for a collection of triangle strips over the polygonal model;
(2) to obtain the simplified versions of the original model;
(3) to store all the information in the data structures.

### 3.1. Data structure

A multiresolution triangle strip is represented by a directed graph with labels in its arcs and a list with the vertices at the beginning of the strip. The class that encodes a multiresolution strip has two fields, namely, a pointer to the list with the vertices at the beginning of the strip (*sBegin*) and a pointer to the graph (*colVertices*), encoded as an adjacency list [3].

Conceptually, each vertex of the strip is identified with a graph node (**class** `ColumNode`) and each inner edge with a graph arc (**class** `RowNode`).

The **class** `ColumnNode` has two fields, that is, an index to the memory address containing the vertex geometric data (*vIndex*) and a pointer to its adjacency list (*neighbours*).

On the other hand, the **class** `RowNode` also has two fields, i.e. an index to the next node in the sequence of vertices that represents the strip (*colIndex*), and an integer that indicates the maximum level of detail at which it is allowed to traverse the arc for the level of detail extraction algorithm.

```
class ColummNode
  unsigned long vIndex;
  RowNode * neighbours;
end class

class MultiresolutionStrip
  RowNode * sBegin;
  ColummNode * colVertices;
end class

class RowNode
  unsigned int colIndex;
  unsigned long res;
end class
```

Fig. 3. Data structures of the MTS model.

The vertex at the beginning of the strip could change as the strip is simplified. The list of strip beginnings indicates, for each level of detail (*res*), which is the vertex at the beginning of the strip (*colIndex*). These two fields have the same meaning as the fields in the **class** `RowNode`, so they can be represented by **class** `RowNode` (see Fig. 3).

#### 3.1.1. Construction example

The construction of a multiresolution strip starts with the original strip. As the strip is simplified, the sequence of vertices that describes it changes. These changes in the sequence of vertices should be introduced into data structures that represent the multiresolution strip.

Let's take the strip in Fig. 4. This figure shows the initial graph and the list with the vertices at the be-

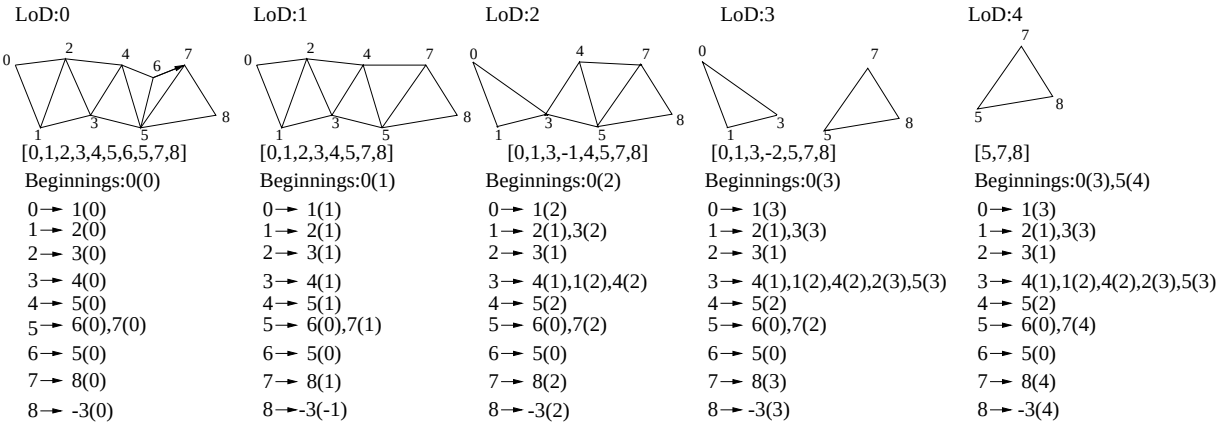


Fig. 4. Example of the construction of a multiresolution strip. (a) Original strip. (b–d) The original strip after three vertex pair contractions. (e) The final multiresolution strip.

ginning of the strip. Let's label the maximum level of detail at which a strip can be represented as level of detail 0 (LoD 0). The sequence of vertices at LoD 0 is 0, 1, 2, 3, 4, 5, 6, −3. The special label −3 specifies that the end of the strip has been reached. After the first pair vertex contraction (2 → 3), LoD 1, the resulting sequence of vertices is 1, 2, 3, 4, 5, 6, 7, −3. The vertex at the beginning of the strip has changed. This change is store in the data structures adding a new element to the list of vertices at the beginning of the strip, its `colIndex` field being initialised with 1 and its res field is initialised with 1. The inner edges that remain at this new LoD update their `res` field to 1 (see Fig. 4).

The second vertex pair contraction (5 → 3), LoD 2, gives the sequence of vertices 1, 2, 3, 4, 3, 6, 7. This sequence represents a generalised triangle strip with a swap operation between vertices 3 and 4. Fig. 4 shows the update graph.

The third vertex pair contraction (3 → 4), LoD 3, gives the sequence of vertices 1, 2, 4, −1, 4, 6, 7. The label −1 indicates that vertex 4 is the end of a sub-strip (the one with sequence 1, 2, 4) and is also the vertex at the beginning of the next sub-strip (the one with sequence 4, 6, 7). Fig. 4 shows how this special node is stored on the graph.

The fourth vertex pair contraction (4 → 6), LoD 4, gives the sequence of vertices 1, 2, 6. The vertex at the end of the strip is vertex 6, so a node with label −3 is

```
{ First we search the strip beginning }
while beginning.res < res and Beginnings ≠ ∅ do
  beginning = next_beginning;
end while
if beginning Not Found then { This strip does not }
  exit
else { exist at this resolution }
  Node = beginnning;
  while not EndStrip do { While there are vertices in strip }
    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;
    end if
    Neighbour = Node.currentNeigbour;
    while Neighbour.res ¡ ResolutionDemmanded do
      Neighbour = nextNeighbour;
    end while
    Node = Neighbour;
  end while
end if
```

Fig. 5. Pseudo-code of the level of detail recovery algorithm.

added at the end of the list of neighbours of node 6 with its `res` field initialised to 4. The inner arcs that remain at this new LoD update their `res` field to 4.

There is another special node labelled −2. This node is used as follows: if, after a pair vertex contraction, the strip is split into two sub-strips, a special node labelled −2 appears between the last node of the first strip and the beginning of the next one.

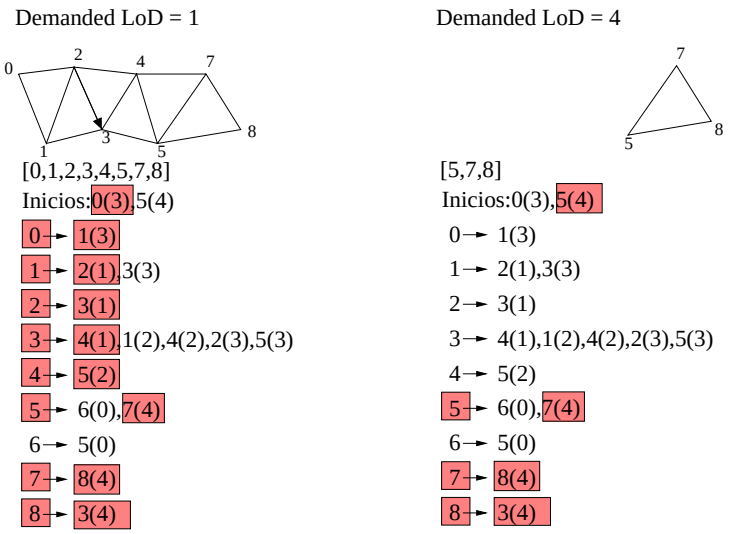


Fig. 6. Two examples of extractions for LoD 2 and LoD 3.

The final graph plus the list with the vertices at the beginning of the strip have all the necessary information to recover the strip at any LoD.

### 3.2. Level of detail recovery algorithm

The LoD recovery algorithm traverses the graph in order to extract the requested LoD. The graph is traversed through those arcs having a label (`res` field of **class** `RowNode`) greater or equal to the requested LoD. These arcs are called compatible arcs.

The algorithm proceeds in two steps:

1. The algorithm determines which vertex at the beginning of the strip is compatible with the requested LoD.
2. The graph is traversed from the vertex at the beginning through those arcs compatible with the requested LoD, until a special −3 node is reached.

The pseudo code of the algorithm is shown in Fig. 5.

Fig. 6 shows two examples of the LoD extraction process. The compatible nodes and compatible arcs are highlighted in grey. The nodes and arcs visited but which are not compatible with the requested LoD are highlighted in light grey.

## 4. Coherence

Coherence in a multiresolution model means to take advantage of the last information extracted prior to the extraction of the newly requested LoD. The use of coherence decreases the time consumed by the recovery algorithm, thus saving the need to repeat calculations that have already been performed.

Each sequence of a multiresolution strip has a LoD interval of validity. The minimum of the interval (`minRes`) is the maximum value of those arcs that are not compatible with the requested LoD. The maximum of the interval (`maxRes`) is the minimum value of the arcs that are compatible with the requested LoD. In this way, if the newly requested LoD is between these two values, the sequence of vertices remains valid. The minimum and maximum of the interval are retrieved when the graph is been traversed. This interval is recalculated every time it becomes invalid.

## 5. Results

The MTS model was subjected to several tests. The purpose of these tests is to evaluate the visualisation time in a real time visualisation application. The results are compared with those of the PM and MOM-Fan models, which use the triangle primitive and the triangle fan respectively, both in the data structure and at the rendering stage. A proper implementation of PM is used and was previously verified [16]. The time obtained in the extraction is the same as that published by the author.

The tests were performed on an HP Kayak XU with two Pentium III processors at 550 MHz and 1 GB. of main memory. A GALAXY graphics card, by Evans & Sutherland with 15 MB. video memory was used.

### 5.1. Model sizes

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/).

Table 1 shows the characteristics of the objects used in the experiments and the sizes of the three models that were compared. The sizes of the MTS model are similar to those of PM and MOM-Fan. The model sizes of the MTS model are smaller than those first published in [2].

Table 1
Characteristics of the objects and model sizes in MB

| Model | #Strips | #Triangles | #Vertices | MTS | PM | MOM-Fan |
|---|---|---|---|---|---|---|
| Cow | 136 | 5804 | 2904 | 0.253 | 0.256 | 0.202 |
| Sphere | 173 | 30624 | 15314 | 1.274 | 1.318 | 1.104 |
| Bunny | 1229 | 69451 | 34834 | 2.964 | 2.993 | 2.245 |
| Phone | 1747 | 165963 | 83044 | 6.766 | 7.144 | 6.940 |

Table 2
Time extraction reduction using coherence in the MTS model (time in milliseconds)

| Model | Linear | | Exponential | | Random | |
|---|---|---|---|---|---|---|
| | Coherence | Without | Coherence | Without | Coherence | Without |
| Cow | 0.469 | 1.519 | 0.389 | 1.713 | 1.560 | 1.640 |
| Sphere | 7.364 | 10.209 | 6.234 | 10.830 | 10.700 | 10.080 |
| Bunny | 15.197 | 27.120 | 13.052 | 29.067 | 27.500 | 27.965 |
| Phone | 44.264 | 65.539 | 38.613 | 70.254 | 65.470 | 66.410 |

### *5.2. Time extraction reduction using coherence*

Table 2 shows the extraction time with and without coherence for the MTS models. This time remains almost constant regardless of the number of LoDs extracted for models without coherence. This is due to the fact that every multiresolution strip is processed for a newly requested LoD. If coherence is used, only multiresolution strips that have changed are processed. If the number of LoDs is large, the distance between them is small. Thus, advantage can be taken of most of the information extracted in the previous LoD.

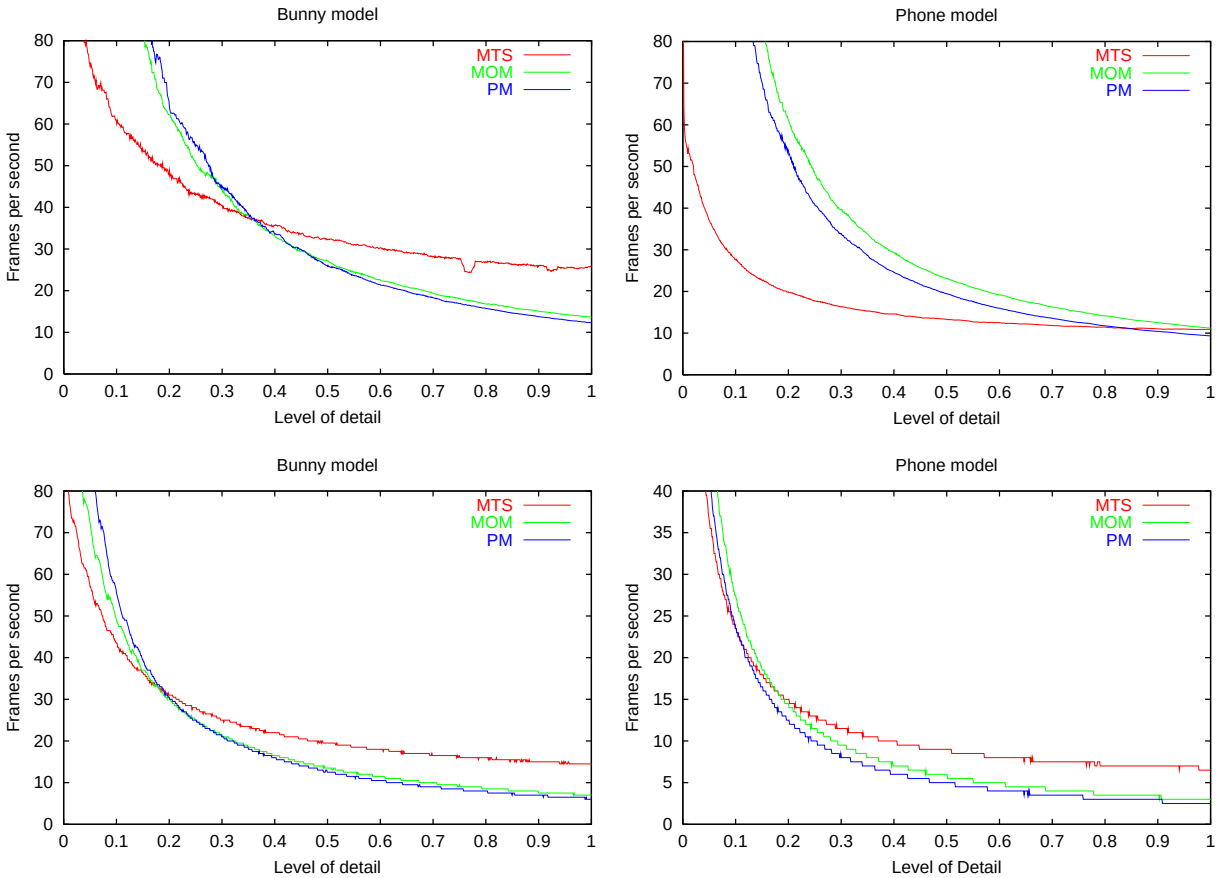


Fig. 6. Visualisation frame rate. Immediate visualisation (up), one extraction per second (down).

Table 3
Comparison between Skip Strips and MTS for rendering (time in milliseconds)

| #Triangles | Skip Strips | | MTS | |
|---|---|---|---|---|
| | Display | #Vertices | Display | #Vertices |
| 10.2K | 16.0 | 22K | 17.8 | 23K |
| 50.8K | 78.1 | 83K | 35.7 | 71K |
| 69.4K | 101.0 | 92K | 40.0 | 83K |

Table 4
Computing Skip Strips per frame vs. MTS (time in milliseconds)

| #Triangles | Skip Strips | | | MTS | | |
|---|---|---|---|---|---|---|
| | Update | Display | #Vertices | Updates | Display | #Vertices |
| 5K | 5.3 | 6.4 | 10K | 5.9 | 13.5 | 11K |
| 30K | 13.1 | 30.3 | 42K | 6.4 | 28.7 | 51K |
| 65K | 25.2 | 85.2 | 90K | 5.8 | 38.4 | 81K |

### *5.3. Visualisation time*

The MTS model was compared with PM and MOM-Fan models. These models were subjected to two kinds of experiments.

The purpose of the first experiment was toevaluate the performance of each model in terms of number of frames per second when a series of LoDs are requested and the object is immediately visualised.

The second experiment is more realistic.The user of a real time application needs to feel smoothe, not jerky, motion. In this experiment a new LoD is requested each second. After the extraction, the model is visualised until a new LoD is requested. Results are shown in Fig. 6.

### *5.4. MTS versus Skip Strips*

Skip Strips is the closest multiresolution model to MTS. In this section, the Skip Strips results that have been published are compared with those for the MTS. Skip Strips has been implemented on an SGI Onyx 2 with four R 10000 processors and 1 GB. RAM, but timings reported do not assume parallelisation. Timings are not comparable because they were obtained on different platforms. The number of vertices sent to the graphics system is, however, platform independent. The only common model is the Stanford Bunny. Comparisons are shown in Tables 3 and 4.

## 6. Conclusions and future work

The main contribution of the multiresolution model presented in this paper is the use of the triangle strip primitive as the basis for the data structure and at the rendering stage. The main benefit in using the triangle strip primitive is the decrease in the amount of data sent to the graphics system, which thus speeds up the rendering stage.

The MTS model has been compared with two modes: PM and MOM-Fan. PM uses the triangle primitive both in the data structure and at the rendering stage. Whereas MOM-Fan uses the triangle fan primitive both in the data structure and at the rendering stage.

The model sizes created with MTS are comparable to those of PM. As regards visualisation time, MTS provides significantly higher frame rates than those offered by PM. This is mainly due to the fact that the MTS model sends fewer vertices to the graphics system than PM. In the first kind of experiment, involving one extraction and its visualisation, MTS reaches a frame rate that doubles the frame rate for PM at high levels of detail. As the model is simplified, PM frame rate approaches that of MTS. This is due to the fact that the level of detail extraction algorithm for MTS is slower than that of PM so the extraction time and the rendering time are almost the same at that level of detail.

The conclusion for the second experiment, one extraction per second, are still better for the MTS model. In this case, the crossing point moves towards a lower

level of detail. Obviously, if the same amount of information is rendered more than once per second the frame rate improves for the MTS model.

The crossing point for the second experiment is reached at about 20% triangles of the original model. In this case, another acceleration technique could be employed, for example, using imposters.

The same reasoning can be applied between the MTS model and MOM-Fan. The main drawback for the MOM-Fan model is that the number of triangles per fan in the model is only three or four so the amount of information sent to the graphics system is still higher than in the case of the MTS model.

The MTS model has also been compared with Skip Strips. The main result is that the number of vertices sent to the graphics system is lower for MTS than for Skip Strips.

The main conclusion is that the use of triangle strips in a multiresolution model provides frame rates far beyond of those provided by models based on the triangle or triangle fan primitive.

At www3.uji.es/∼belfern/Java/MTS.htmlcan be found a java implementation of the MTS model.

The main drawback of the MTS model is the high extraction time. Introducing coherence inside every strip is something that must be address in future work in order to reduce the extraction time. View-dependent visualisation and progressive transmission of the models through a computer network are also lines of future work.

## Acknowledgments

This research was supported by grant P1.1B2001-09 (Fundació Caixa Castelló - Bancaixa).

## References

[1] E.M. Arkin, M. Helod, J.B.S. Mitchell, S.S. Skiena, Hamiltonian triangulation for fast rendering, Vis. Comput. 12 (9) (1996) 429–444.
[2] Ó. Belmonte, I. Remolar, J. Ribelles, M. Chover, C. Rebollo, M. Fernández, Multiresolution modelling using connectivity information, J. WSCG 10 (1) (2002) 71–78.
[3] G. Brassard, P. Bratley, Fundamentals of Algoritmics, Prentice Hall, 1996.
[4] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Comput. Graph. Forum (1998) 167–174.
[5] J.H. Clark, Hierarchical Geometric models for visible surface algorithms, Comun. ACM 19 (10) (1976) 547–554.
[6] J. El-Sana, F. Evans, A. Varshney, S. Skiena, E. Azanli, Efficient computing and updat-ing triangle strips for real-time rendering, J. Comput. Aided Des. 32 (13) (2002) 753–772.
[7] F. Evans, S. Skiena, A. Varshney, Optimising triangle strips for fast rendering, IEEE Visualisation ’96 (1996) 319–326, http://www.cs.sunysb.edu/stripe/.
[8] M. Garland, P. Heckbert, Surface simplification using quadric error metrics, in: Proceedings of SIGGRAPH’97, 1997, pp. 209–216.
[9] M. Garland, P. Heckbert, Survey of polygonal surface simplification algorithms, Mul-tiresolution Surface Modeling Course Notes of SIGGRAPH’97, 1997, http://www.cs.cmu.edu/garland/quadrics/.
[10] M. Garland, Multiresolution modelling: survey and future opportunities, State-of-the-Art Reports of EUROGRAPHICS ’99, (1999), 111–131.
[11] H. Hoppe, Progressive meshes, in: Proceedings of SIGGRAPH ’96, 1996, pp. 99–108.
[12] H. Hoppe, View-dependent refinement of progressive meshes, in: Proceedings of SIG-GRAPH ’97, 1997, pp. 189–197.
[13] P. Lindstrom, Model Simplification using Image and Geometry-Based Metrics, M.S. Thesis, Georgia Institute of Technology, 2000.
[14] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, R. Huebner, Level of Detail for 3D Graphics, Elsevier Science, 2003.
[15] E. Puppo, R. Scopigno, Simplification, LOD and Multiresolution—Principles and Applications, Tutorial Notes of EUROGRAPHICS’99, 1999.
[16] J. Ribelles, M. Chover, A. López, J. Huerta, A First Step to Evaluate and Compare Mul-tiresolution Models, Short Papers and Demos of EUROGRAPHICS’99, 1999, 230–232.
[17] J. Ribelles, A. López, I. Remolar, O. Belmonte, M. Chover, Multiresolution modelling of polygonal surface meshes using triangle fans, Lecture notes in computer science 1953, in: Proceedings of the Ninth Discrete Geometry for Computer Imagery Conference, 2000, pp. 431–442.
[18] M. Woo, J. Neider, T. Davis, D. Shreiner, OpenGL Programming Guide, Addison-Wesley, 1999.

**Ó. Belmonte** is an assistant professor at the department of computer languages and systems (Universitat Jaume I in Castelló—Spain) where he teaches computer graphics and multimedia. He received his MSc in physics in 1992 and he obteined his PhD in computer science from the Universitat de Valéncia (Spain) in 2002. His research areas include multiresolution modelling, real-time visualisation and Java technology. He is a member of Eurographics.

**I. Remolar** received her MSc in computer science from the Universidad Politécnica de Valencia, Valencia, in 1995 and is currently pursuing her PhD in computer science from Universitat Jaume I, Castelló. Since 2000 she has been an assistant professor at the department of computer languages and systems at Universitat Jaume I, where she teaches computer graphics. Her research areas include multiresolution modelling, real-time visualisation of trees and terrain. She is a member of Eurographics.

**J. Ribelles** received his MSc in computer science from the Universidad Politécnica de Valencia, Valencia, in 1993 and his PhD in computer science from Universitat Jaume I, Castelló, in 2000. He has been an assistant professor at the department of computer languages and systems at Universitat Jaume I, where he teaches computer graphics sice 1997. He was a visiting computer scientist in the computer science department of Carnegie Mellon University in 1999, working in the areas of geometric modelling and surface simplification. His research areas include multiresolution modelling, real-time visualisation and image-based modelling and rendering. He is a member of Eurographics.

**M. Chover** received his Msc in computer science from the Universidad Politécnica de Valencia in 1992, and his PhD in computer science from the Universidad Politécnica de Valencia in 1996. Since 1992, he has been an assistant professor at the department of computer languages and systems at Universitat Jaume I, Spai, where he teaches computer science. His research areas include multiresolution modelling, real-time visualisation and collaborative virtual worlds. He is a member of Eurographics.

**M. Fernández** is an assistant professor at the department of computer science (Universitat de Valéncia, Spain) and the coordinator of the ARTEC Graphics Group at the Robotics Institute. He obtained his MSc in physics in 1992 and he recieved his PhD in computer science from the Universitat de Valéncia. He has been working in the field of real time computer graphics and virtual reality since 1992. His PhD was oriented towards the generation of traffic for driving simulators. Currently, he is interested in the area of highly immersive virtual environments and he is involved in the development of an immersive visualisation facility call VISIONARC.