Multiresolution Modelling of Polygonal Surface Meshes Using Triangle Fans*

José Ribelles, Angeles López, Inmaculada Remolar,
Oscar Belmonte, and Miguel Chover

Departamento de Informática, Universitat Jaume I,
E-12080 Castellón, Spain
{ribelles,lopeza,remolar,belfern,chover}@uji.es

Abstract. Multiresolution modelling of polygonal surface meshes has been presented as a solution for the interactive visualisation of scenes formed by hundreds of thousands of polygons. On the other hand, it has been shown that representing surfaces using sets of triangle strips or fans greatly reduces visualisation time and provides an important memory savings. In this paper we present a new method to model polygonal surface meshes. Like the previously explained *Multiresolution Ordered Meshes* (MOM), this method permits the efficient management of an ample range of approximations of the given model. Furthermore, this method utilises the triangle fan as its basic representation primitive. Experiments realised with data sets of varying complexity demonstrate reduced storage space requirements, while retaining the advantages of MOMs.

1 Introduction

One of the principle objectives of multiresolution modelling [1] is to permit interactive visualisation of surfaces formed by thousands of polygons. Given a mesh, M, a multiresolution model defines how to store and retrieve n different approximations or levels of detail (LOD), $M_0, M_1, ..., M_{n-1}$, in an efficient manner.

Several mechanisms have been developed to accelerate the process of visualising polygonal models. For example, strips and fans of triangles (see figure 1) appear as drawing primitives in some graphics libraries, such as *OpenGL*. This type of primitives allow for rapid visualisation. To draw a fan of n triangles, for example, it is only necessary to pass *n+2* vertices, instead of *3n*, to the graphics processor. This not only reduces computation time due to a reduction in vertices, but also an important memory savings.

Obtaining the optimal set of strips or fans for a given surface is a process realised off-line when working with static models. However, when working with a multiresolution model, the surface connectivity changes with changes in level
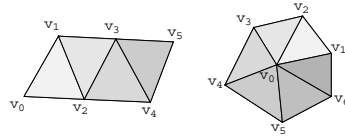

Fig. 1. Example of a strip and a fan of triangles. On the left, a strip defined by $v0, v1, v2, v3, v4, v5$ and on the right, a fan defined by $v0, v1, v2, v3, v4, v5, v6, v1$
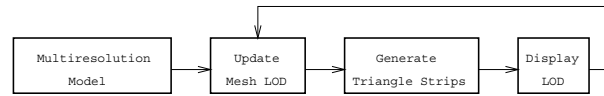

Fig. 2. Visualisation process of a LOD in *VDPM*

of detail, probably with each frame, therefore requiring the dynamic generation of the strips or fans.

This article presents a new multiresolution scheme permitting the recuperation of a level of detail, directly as a set of triangle fans. It is based on the *Multiresolution Ordered Meshes (MOM)* model presented earlier [2]. The new scheme, called *MOM-Fan*, defines a new data structure and a new traversal algorithm which optimises the triangle fans for visualisation at the LOD required.

Notation. The geometry of a triangulated model, M, is denoted as a tupla $\{\mathcal{V}, \mathcal{F}\}$, where $\mathcal{V}$ is a set of N positions $v_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $1 \leq i \leq N$, and $\mathcal{F}$ is a set of triples $\{j, k, l\}, j, k, l \in \mathcal{V}$, specifying positions of triangles faces.

2   Previous Work

Hoppe [3] presents a multiresoution model, *VDPM*, based on a hierarchical structure of vertices built from a sequence of contractions of edges. The level of detail is determined from a series of criteria based on the view frustum, surface orientation, etc. Changes in these conditions trigger changes in the required LOD, and it is proposed that triangle strips be generated using a greedy algorithm once the component triangles are determined for that LOD (figure 2).

El-Sana et al. [4] presents a data structure, *Skip-Strips*, that maintains triangle strips even though the LOD may change. A *Skip-Strip* is built at run time, from the multiresolution model. At the same time the triangle strips from the original model are obtained. Each time the level of detail changes, the *Skip-Strip* structure is updated based on the required LOD, which permits the update of the strips and their subsequent visualisation (figure 3). As the strips are generally quite short, it has been proposed that they be concatenated previous to visualisation.

The proposal in this article simplifies the LOD visualisation scheme. The multiresolution model itself is encoded using triangle fans, and therefore it is
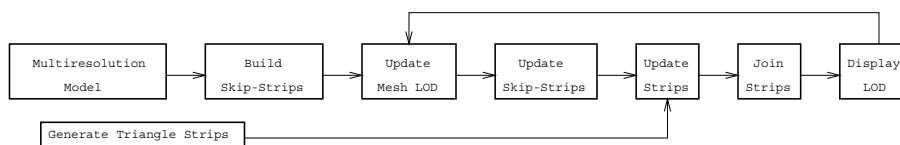
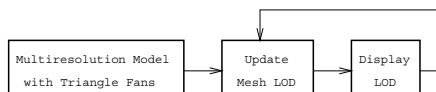**Fig. 3.** Visualisation process of a LOD in *Skip-Strips*

Fig. 4. Visualisation process of a LOD using *MOM-Fan*

necessary to make some adjustments to each fan according to the LOD (figure 4). Figure 9 illustrates three LODs of a *MOM-Fan* object, where the triangles composing each fan have been coloured alike and the boundaries have been highlighted.

Other multiresolution models exist which do not make use of either triangle fans or strips; see [1] for a recent survey of these.

2.1   Review of Multiresolution Ordered Meshes

Multitresolution Ordered Meshes was presented with the idea of improving the interactive visualisation of complex polygonal surfaces. Later, it was extended to exploit frame-to-frame coherence [5]. This permitted the acceleration of LOD recovery, while not affecting visualisation time. Finally, in [6] *MOM* was compared against *Progressive Meshes* [7].

Let M and M^r be the original and multiresolution meshes, respectively. M^r explicitly stores the vertices $\mathcal{V}^r$, and the faces $\mathcal{F}^r$, utilised to represent any resolution:

$$M^r = \{\mathcal{V}^r, \mathcal{F}^r\} \tag{1}$$

To build M^r with n levels of detail, we apply $n-1$ iterations of a simplification method. Each simplification S_i, $0 \leq i < n-1$, produces a new level of detail M_{i+1} and may be represented by the tuple $S_i = \{V_i, F_i, V_i', F_i'\}$ where V_i and F_i are the sets of vertices and faces which are eliminated from M_i, and V_i' and F_i' are the sets of vertices and faces which are added to M_i to gerenerate, finally, M_{i+1}. Therefore, we may express the resulting object, M_{i+1}, as:

$$M_{i+1} = (M_i - \{V_i, F_i\}) \cup \{V_i', F_i'\}, \ 0 \leq i < n - 1 \tag{2}$$

Given that M^r stores all vertices and faces that can be used at any set level of detail, M^r can be defined as:

$$M^r = \bigcup_{i=0}^{n-1} M_i, \; n \geq 1 \tag{3}$$

From the equations 1 - 3, we derive that M^r can be expressed as the initial mesh, $M = M_0$, plus all vertices and faces generated in each iteration of the simplification process:

$$\mathcal{V}^r = \mathcal{V}_0 \cup V_0' \cup V_1' \cup ... \cup V_{n-2}' = \mathcal{V}_0 \cup \bigcup_{i=0}^{n-2} V_i' \tag{4}$$

$$\mathcal{F}^r = \mathcal{F}_0 \cup F_0' \cup F_1' \cup ... \cup F_{n-2}' = \mathcal{F}_0 \cup \bigcup_{i=0}^{n-2} F_i' \tag{5}$$

or also, as the mesh corresponding to the worst level of detail, M_{n-1}, plus the vertices and faces eliminated in each iteration of the simplification process:

$$\mathcal{V}^r = V_0 \cup V_1 \cup ... \cup V_{n-2} \; \cup \; \mathcal{V}_{n-1} = \bigcup_{i=0}^{n-2} V_i \; \cup \; \mathcal{V}_{n-1} \tag{6}$$

$$\mathcal{F}^r = F_0 \cup F_1 \cup ... \cup F_{n-2} \; \cup \; \mathcal{F}_{n-1} = \bigcup_{i=0}^{n-2} F_i \; \cup \; \mathcal{F}_{n-1} \tag{7}$$

The basic idea of *MOM* is based on the expressions of the two previous equations. That is, store in ordered form the sequences of vertices and faces eliminated, V_i and F_i, $0 \leq i < n - 2$, plus the vertices and faces corresponding to the worst level of detail $M_{n-1} = \{\mathcal{V}_{n-1}, \mathcal{F}_{n-1}\}$. Each stored face is identified by a value representing its position in the face sequence ordered according to equation 5 above. *MOM-Fan* is based on the same idea, the difference being that here we store and manipulate triangle fans instead of isolated triangles. In section 3 we show how to store the fans in the data structure, and in section 4 how to recover those fans which form a given level of detail.

3   Data Structure

The data structure presented here is based on a list of lists similar to that described in [2]. The fundamental difference is the list which stores vertex sequences, which substitutes the anterior list of faces. In figure 5 the data structure of the model is shown. The data structure is formed of three lists:

- *Vertex list* (*vertexList* field). Stores the vertices of the mesh in an ordered fashion according to their elimination in the simplification process. Each represents the initial vertex of a fan and consists of its co-ordinates (*coord* field) and a pointer to the second vertex in the fan (*secondFanVertex* field).

```
                          struct ControlLod
struct Vertex                int deletedVertices;
  float coord[3];            int generatedFaces;
  int secondFanVertex;    end struct
end struct
                          struct MOM-Fan
struct FanVertex             int nVertices, nFanVertices, nLods;
  int id;                    struct Vertex vertexList[];
  struct Vertex *v;          struct FanVertex fanVertexList[];
end struct                   struct ControlLod controlLodList[];
                          end struct
```

Fig. 5. *MOM-Fan* Data Structure

– *Fan list* (*fanVertexList* field). Stores the fans as vertex sequences (except the initial vertex, already stored in the vertex list). For each fan we store the vertices of the triangles which disappear when the common vertex is eliminated. Each stored vertex consists of a pointer to the vertex in the vertex list (*v* field) and an identifier (*id* field). The identifier references the face of the fan represented by that vertex. The fans are also ordered according to their elimination in the simplification process.
– *Control List* (*controlLodList* field). For each LOD, this list stores the information necessary to recover the data pertaining to that LOD.

3.1 Construction Process

The process of constructing the model is divided into 2 steps. The first involves the tasks which must be repeated during each iteration of the simplification. The second includes a group of operations which finish building the multiresolution model M^r. To simplify the explanation let us assume that each simplification realised eliminates only one vertex. In figure 6 we show the mesh to be utilised in explaining the construction, initialisation, the result of the first stage -eliminating vertices $v4$ and $v1$- and the result of the second stage.

Initialisation. Before beginning this process it is necessary to initialise M^r with M_0, as it is the first level of detail. For this it is necessary to update *controlLodList* with data from the mesh M_0, filling in the fields *deletedVertices= 0* and *generatedFaces= $|\mathcal{F}_0|$*. The vertex list and fan list are empty.

First stage. For each iteration in the simplification process the following tasks are realised:

1. Store in *fanVertexList* the vertices which form the fan, except for the vertex eliminated (common vertex).
2. Store in *vertexList* the eliminated vertex and place the pointer at the first of the remaining vertices of the fan.
3. Update *controlLodList* with the deleted vertices and the total number of generated faces.
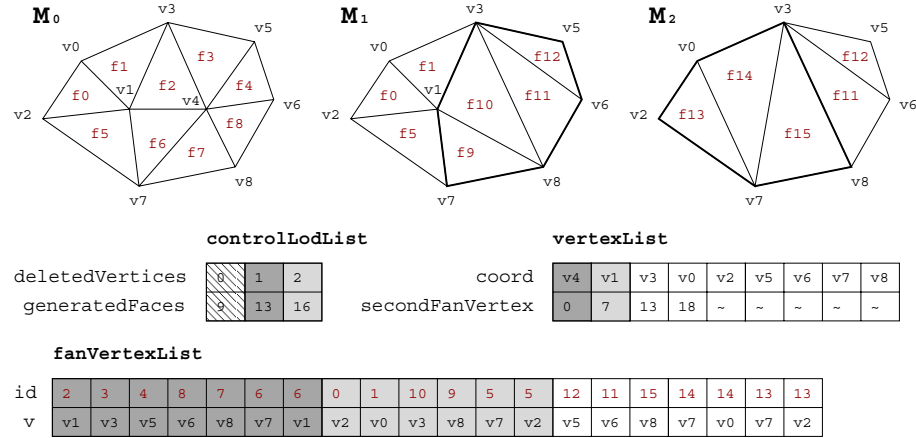
Fig. 6. Example of model construction. Shaded cells indicate initialisation. In dark grey, data added after the elimination of $v4$, and in light grey, data added after the elimination of $v1$, both in the first stage. In white, data added as a result of the second stage

Second stage. To complete the data structure:

1. Add the mesh containing the worst LOD, as a set of fans, in the same manner as in the first stage. The vertices that do not have an assigned fan are put at the end of the list.
2. Update the pointers of the fan vertices so that they point to the vertices in the new vertex list.

With this step we conclude the construction of the data structure, which we can generalise as being the elimination of more than one vertex in the simplification process similar to that in [2].

3.2   Storage Cost

Assuming that the cost of storing a real, an integer and a pointer is one word, and that we store three lists, one for each data type in the model (see figure 5), the total storage cost will be $4|\mathcal{V}^r| + 2|\mathcal{F}_a^r| + 2n$ words, being $|\mathcal{F}_a^r|$ the size of *fanVertexList*. The order of $|\mathcal{V}^r|$ (see eq. 4) in the worst case is quadratic with regard to $|\mathcal{V}_0|$. The best case appears when we use a simplification method based on vertex decimation [8], where $|\mathcal{V}^r| = |\mathcal{V}_0|$ and may reach $|\mathcal{V}^r| = 2|\mathcal{V}_0|$ in the case of simplification methods based on elimination of edges [9].

With regard to $|\mathcal{F}_a^r|$, in the worst case, the cost also is of quadratic order with respect to $|\mathcal{F}_0|$. If we use a simplification method based on vertex decimation and we assume that the number of faces around a given vertex to be an average of 6 and that $|\mathcal{V}_i| \approx |\mathcal{F}_i|/2, 0 \leq i < n-1$, we have $|\mathcal{F}_a^r| \approx 3.5|\mathcal{F}_0|$. With a method based on elimination of edges, $|\mathcal{F}_a^r| \approx 5.5|\mathcal{F}_0|$.

```
for each vertex v from DV and while there exist triangles to paint
  InterruptionFan <- true {force the a.1 case}
  for each vertex vᵢ of the fan associated with v, from the first to the penultimate
    if fanVertexList[vᵢ].id < GF then {case a) paint vᵢ}
      if InterruptionFan then
        paint(v) {case a.1) paint the initial vertex}
        InterruptionFan <- false {do not repeat case a.1)}
      end if
      paint(vᵢ)
      NF <- NF-1
    else {case b) do not visualise vᵢ}
      if no InterruptionFan then
        paint(vᵢ) {paint vᵢ to close triangle}
        InterruptionFan <- true {force case a.1)}
      end if
    end if
  end for
  if no InterruptionFan then
    paint(vᵢ) {last vertex vᵢ from the list, to close fan}
  end if
end for
```

Fig. 7. Visualisation algorithm

4   Algorithm for Visualising a LOD

The visualisation algorithm of a LOD, which is shown in figure 7, requires the initial calculation of the number of triangles to visualise at that LOD. Afterward, it traverses the data structure recovering the triangle fans pertaining to the required LOD.

Given a LOD k in the control list we store the number of vertices not pertaining to k, DV, and the number of generated faces, GF. Based on these data it is trivial to obtain the number of faces, NF, which pertain to k:

```
DV= controlLodList[k].deletedVertices;
GF= controlLodList[k].generatedFaces;
NF= GF-(vertexList[DV].secondVertexFan- DV)
```

Using the example in figure 6, suppose we wish to visualise the best LOD (M_0). The fans to be generated, ordered by the appearance of vertices in the data structrure, are: (v4 v1 v3 v5 v6 v8 v7 v1) which correspond to the faces (f2 f3 f4 f8 f7 f6); (v1 v2 v0 v3) and (v1 v7 v2), which correspond to the faces (f0 f1) and (f5), respectively. The first vertex of the fan always corresponds to the eliminated vertex. The remaining vertices are stored in *fanVertexList*, and are associated with the eliminated vertex. For each fan only the t first vertices of $t+1$ vertices composing it are processed, because each of them represents one of the t faces represented whereas the last vertex serves only to complete the fan.

However, it may occur that some vertices do not pertain to the required LOD. In the example, because f10 and f9, implicitly associated to v1, should not be

visualised, there is a jump from vertex v3 to v7. To resolve these jumps without splitting fans, it is necessary to introduce the vertex v1 between v3 and v7, thus producing two degenerate triangles. While the first fan is resolved without degenerate triangles (v4 v1 v3 v5 v6 v8 v7 v1), the second fan (v1 v2 v0 v3 v1 v7 v2) includes two of them, (v1 v3 v1) and (v1 v1 v7). Upon processing each of the vertices two things may occur: a) the vertex identifier indicates that the triangle should be painted, or b) the vertex identifier indicates that the triangle should not be painted. When the second case occurs, this signifies an interruption in the fan, and if this continues further along it is necessary to insert the initial vertex. Therefore, the first time that case a) is encountered after an interruption the insertion should be realised (case a.1 in the algorithm).

The computational cost of the algorithm to extract LOD k depends on the total number of vertices not eliminated, which is at most $|\mathcal{V}^r| - k$, and the total number of vertices of the associated fans, which is at most $|\mathcal{F}_a^r| - 2k$. $|\mathcal{V}^r|$ and $|\mathcal{F}_a^r|$ are, in the worst case, of quadratic order with respect to $|\mathcal{V}_0|$ and $|\mathcal{F}_0|$, respectively, but this case differs substantially from the normal case. With the method of simplification by elimination of vertices the cost is $O(8|\mathcal{V}_k|)$ and with a method based on edge elimination, $O(13|\mathcal{V}_k|)$.

5   Results

The experiments were realised utilising a Silicon Graphics RealityEngine 2, with a MIPS R10000 at 194 MHz and 256 Mb RAM. Coding of the model was in $C++$ and utilised *OpenGL* as its graphics library. The simplification method used to construct the multiresolution representations is that proposed by Garland and Heckbert [9] based on contraction of edges. The meshes come from the *Stanford University Computer Graphics Laboratory* (http://www-graphics.stanford.edu/data/3Dscanrep/) and *Cyberware* (http://www.cyberware.com/models/).

In table 1 we summarise the characteristics and storage costs of the objects used in the experiments. For each of them we indicate the number of vertices and faces of the original model, and its storage cost assuming a structure based on a vertex list and a triangle list [10]. Also it is assumed that a word (integer, real, or pointer) carries a set cost of 4 bytes. With regard to the multiresolution ordered mesh (MOM) representation, we indicate the number of levels of detail, the number of faces and the total storage cost. Regarding the new representation proposed in this article, we indicate the number of *fanVertex* and the total storage cost. It can be observed that the number of fan vertices stored in the new representation is higher than that of the faces in the MOM representation. However, given that the cost to store a fan vertex is less than for a face, the storage cost of the new list provides a memory savings of about 20% over the face list. The repercussion of this is that the total storage cost is reduced by an important amount, approximately 15%, due to the fact that the list which is reduced is the "heaviest" list in the model (compare the number of faces with the number of vertices and LODs).

Table 1. Characteristics and storage costs

	Original			MOM			MOM-Fan	
	Vertices	Faces	MB	Lods	Faces	MB	Fan V.	MB
Cow	2,905	5,804	0.100	2,803	14,982	0.237	17,852	0.202
Sphere	15,315	30,624	0.526	15,264	83,486	1.306	98,793	1.104
Bunny	34,835	69,451	1.193	33,990	182,192	2.876	216,759	2.445
Phone	83,045	165,963	2.850	81,668	441,181	6.940	523,844	5.887
Isis	187,871	375,736	6.450	187,370	993,559	15.667	1,181,373	13.309
Buda	543,653	1,085,636	18.646	543,106	2,754,083	43.957	3,297,593	37.598

In figures 8(a) and 8(d) the behavior of the new representation is shown, using the *Bunny* and *Buda* models (some views of them are shown in figures 10 and 11). On the X-axis the level of detail is represented, where 0 is the poorest and 1 the best. On the Y-axis we show the time spent (in seconds) for the model to recover the data pertaining to a given LOD and to visualise them. The behavior is similar to that obtained with the earlier MOM scheme and one can observe the linear response with respect to the number of triangles of the LOD visualised. The improvement gained by the use of triangle fans is diminished somewhat by the slight increase in data recovery time due to a more complex algorithm, the short length of fans (an average of 3.3 triangles per fan), the degenerate triangles, and the overhead caused by executing, for each fan, the instructions *glBegin* and *glEnd* in the *OpenGL* implementation. In figures 8(b) and 8(e) we show the number of non-degenerate triangles and the number of degenerates (around 23% of the total triangles) sent to the graphic subsystem per LOD. In figure 8(c) and 8(f) we show the number of vertices sent per LOD in the MOM representation, and those sent in the new representation (about 40% fewer).

6   Conclusions and Future Work

In this article we present a new multiresolution scheme which permits the storage and visualisation of the distinct levels of detail as triangle fans. The objective is double: to reduce the visualisation time and also the space (storage) cost.

The experimental results show a reduction of about 15% in storage cost with respect to the previous MOM representation, upon which the new approximation is based. However, the behavior of the 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.

This work will proceed from this moment on, toward the utilisation of triangle strips which, in principle, will permit a higher average number of triangles per strip than has been obtained using fans. In this manner we expect the storage cost to be further improved, as well as visualisation times.
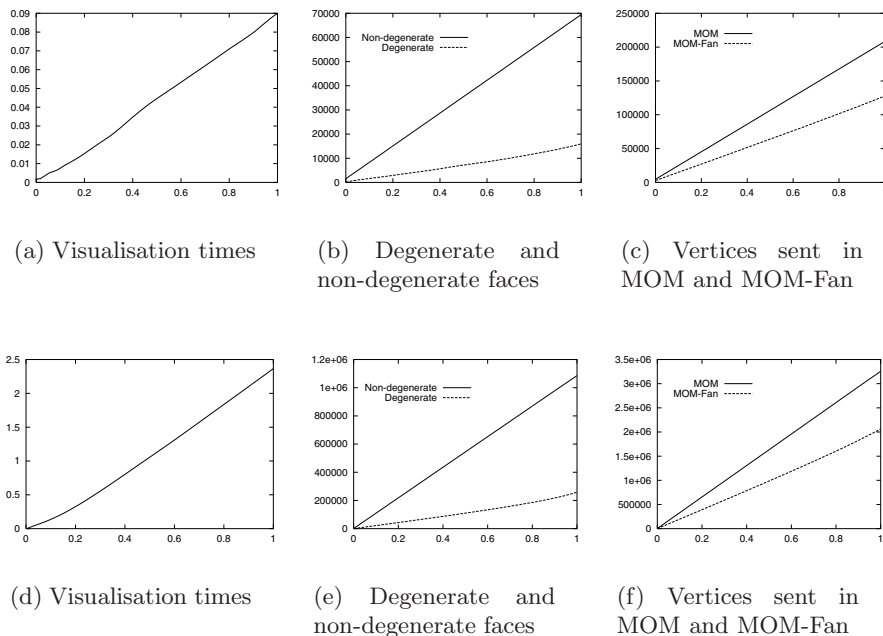
Fig. 8. Results: a), b) and c) *Bunny* model; d), e) and f) *Buda* model

References

1. Garland, M.: Multiresolution Modeling: Survey & Future Opportunities. State of the Art Reports of EUROGRAPHICS '99 (1999) 111–131 431, 433
2. Ribelles, J., Chover, M., Huerta, J., Quiros, R.: Multiresolution Ordered Meshes. Proc. of 1998 IEEE Conference on Information Visualization (1998) 198–204 432, 434, 436
3. Hoppe, H.: View-Dependent Refinement of Progresive Meshes. Proc. of SIG-GRAPH'97 (1997) 189–198 432
4. El-Sana, J., Azanli, E., Varshney, A.: Skip Strips: Maintaining Triangle Strips for View-dependent Rendering. Proc. of IEEE Visualization 1999 (1999) 131–137 432
5. Ribelles, J., Chover, M., Lopez, A., Huerta, J.: Frame-to-frame Coherence of Multiresolution Ordered Meshes. Proc. of CEIG'99 (1999) 91–104 433
6. Ribelles, J., Chover, M., Lopez, A., Huerta, J.: A First Step to Evaluate and Compare Multiresolution Models. Short Papers and Demos of EUROGRAPHICS'99 (1999) 230–232 433
7. Hoppe, H.: Progressive Meshes. Proc. of SIGGRAPH'96 (1996) 99–108 433
8. Schroeder, W. J., Zarge, J. A., Lorensen, W. E.: Decimation of Triangle Meshes Proc. of SIGGRAPH'92 (1992) 65–70 436
9. Garland, M., Heckbert, P.: Surface Simplification Using Quadratic Error Metrics. Proc. of SIGGRAPH'97 (1997) 209–216 436, 438

10. Foley, J. D., van Dam, A., Feiner, S., Hughes, J., Phillips, R.: Computer Graphics. Principles and Practice. Addison-Wesley Publishing Company (1990)   438
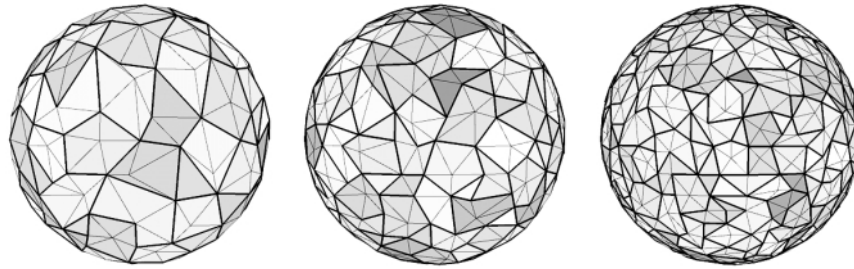

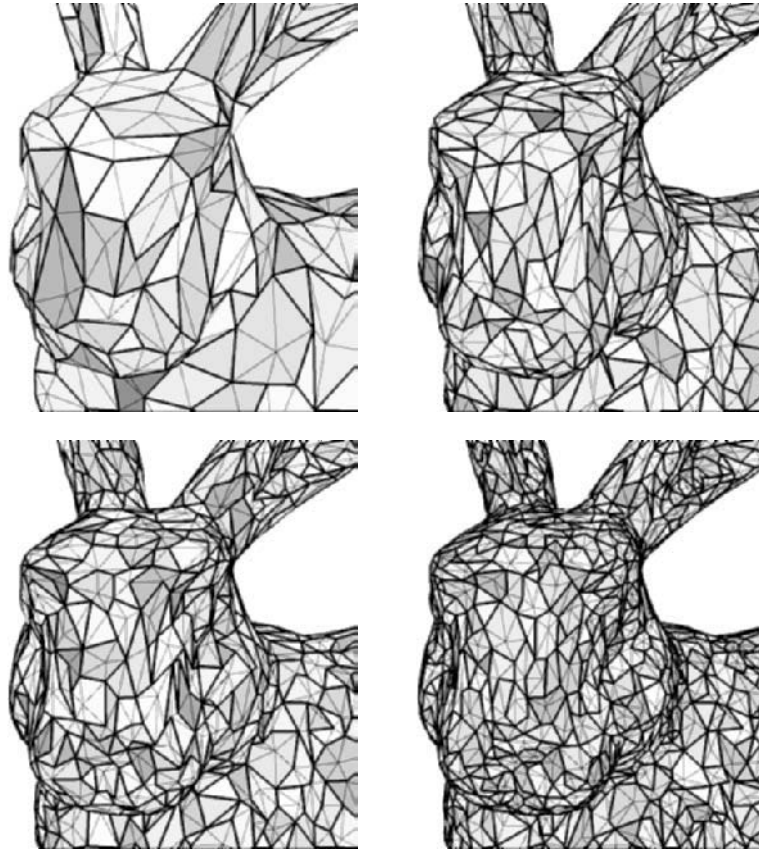**Fig. 9.** Three levels of detail of the *Sphere* model visualised using fans

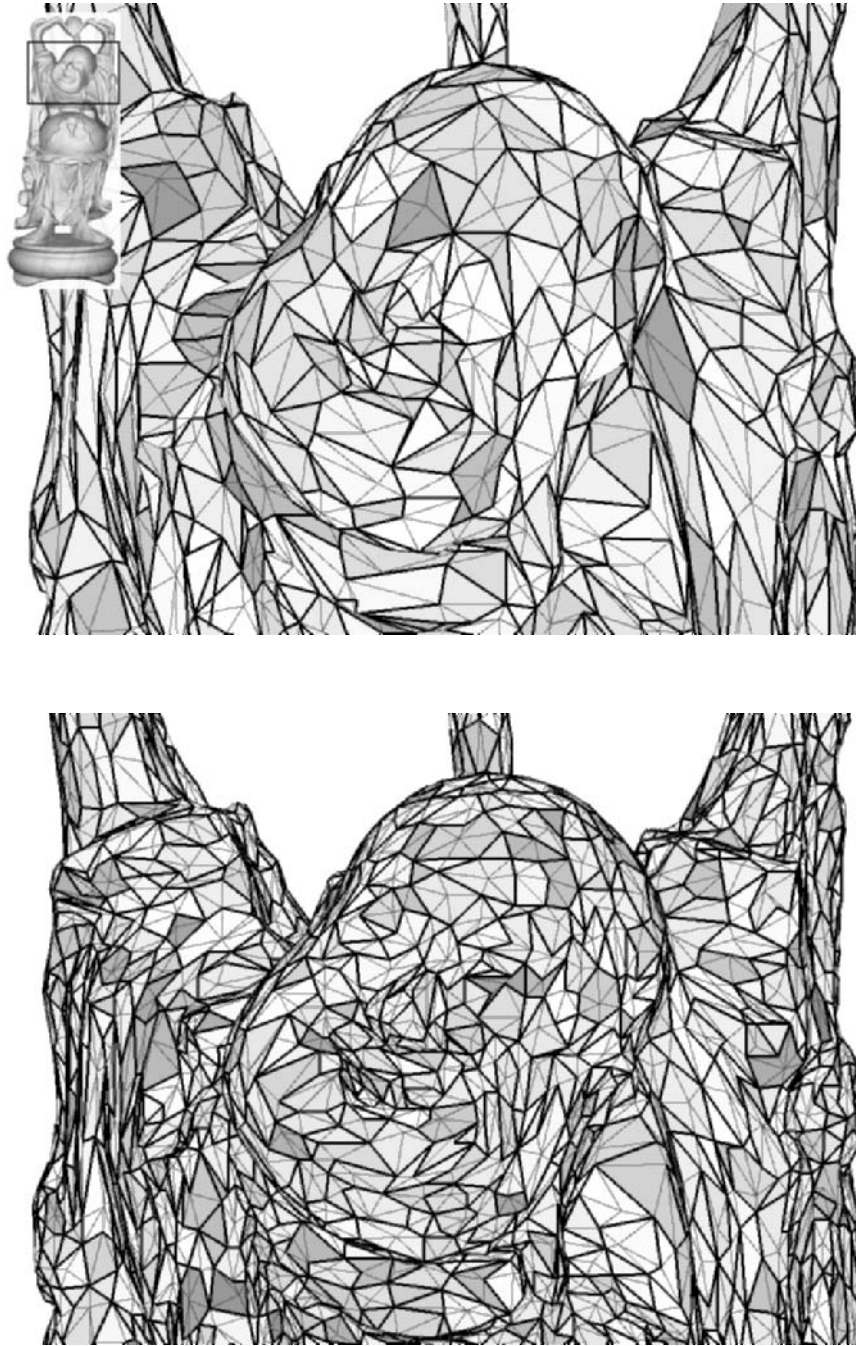**Fig. 10.** Four levels of detail of the *Bunny* model visualised using fans

Fig. 11. Two levels of detail of the *Buda* model visualised using fans