Crossroads
The ACM Student Magazine

Levels of Detail & Polygonal Simplification

by ***Mike Krus,***
Patrick Bourdot,
***Françoise Guisnel**, and*
Guillaume Thibault

Abstract

This paper covers the techniques of *Polygonal Simplification* in order to produce *Levels of Detail* (LODs). The problem of creating LODs is a complex one: how can simpler versions of a model be created? How can the approximation error be measured? How can the visual degradation be estimated? Can all this be done automatically? After exposing the basic aims and principles of polygonal simplification, we compare recent algorithms and state their various qualities and weaknesses.

Introduction

The computation and storage requirements for scenes such as those used in virtual reality applications far exceeds the capacity of modern graphics hardware. These scenes have a complex structure and their display requires a huge number of polygons, even when considering only the portion of the scene that is visible for a given frame. As early as 1976, Clark suggested using simpler versions of the geometry for objects that had lesser visual importance, such as those far away from the viewer [5]. These simplifications are called **Levels of Detail** (LODs).

We do not cover here the actual conditions in which these LODs might be used and will only briefly mention the methods used for choosing a LOD at display time. Briefly, these simplifications may be used when the object appears to be small [11], when it is moving, and when it is in the observer's peripheral vision [25]. Reddy evaluates the support for LOD management in various virtual reality software packages [19].

We cover here only the algorithms which produce LODs filtering the geometry to produce a model with fewer polygons. Other techniques may be used [13], such as hierarchical structures [4,17] and impostors [18]. The most recent algorithms are presented here. We compare their choices in heuristics and computation and state their relative strengths and weaknesses. Other surveys may be also read [10,24]. We first study the basic concepts involved in levels of detail generation. Then, we discuss the techniques involved in polygonal simplification and present six recent algorithms. Finally, we consider problems that should be further investigated.

Levels of Details

The aim of polygonal simplification, when used for levels of detail generation, is to remove primitives from an original mesh in order to produce simpler models which retain the important visual characteristics of the original object. Ideally, the result should be a whole series of simplifications (as shown in Figure 1, made using [6]), which can be used in various conditions. The idea, in order to maintain a constant frame rate, is to find a good balance between the richness of the models and the time it takes to display them.

Figure 1 - Using LODs for distant objects

Algorithms require some kind of heuristics to choose the relevant primitives and control the simplification. We present here various additional criteria that simplification algorithms should consider. Algorithm identifiers (A1, ..., A6) refer to the number of the algorithm in the following section.

LOD Continuum

As stated earlier, we need a series of simplifications to choose from at display time. If this list is *continuous*, i.e. two successive LODs differ by only one or two polygons, it is called a **geomorph** (see **Figure 2**, taken from [**9**]). Algorithms **A1** and **A2** produce data structures from which any simplification can be retrieved. Such data structures enable the smooth display of LODs and the change from one to the next is not noticeable.

Figure 2 - A succession of LODs

The most common practice is to produce a limited number of LODs. Because differences between LODs are thus greater, the switching from one to the other is usually noticeable by the observer (this is known as the **popping effect**). The main difficulty in this case is to figure out which levels of simplification might be needed in the scene.

Shape Preservation

The simplification must be conducted in such a way that the general shape and characteristics which make the object easily identifiable are preserved. Thus, the algorithms have to look for some distinct features of the object:

- **Planar area** can be identified by inspecting the normals of adjacent polygons. These polygons can then be merged to form bigger ones. This is the most common type of simplification as it is relatively easy to compute (although the adjacency relationship must be known).
- **Sharp edges** may appear and should be preserved. They can be found by comparing the angle between the normals of adjacent faces. They can then be simplified by merging connected edges which are nearly colinear.
- **Pointed edges** (such as the tip of a pyramid) must be preserved since they have many chances of appearing on the silhouette of the object. They can be detected by measuring the local curvature around a vertex. They can be simplified by pushing neighboring vertices towards the tip of spike.

Most algorithms look for sharp edges (**A2**, **A4**, **A6**), colinear connected edges (**A6**), and coplanar adjacent faces (**A2**, **A3**, **A4**, **A5**, **A6**).

But when testing, for example, for coplanar faces, the algorithms have a threshold value for the angle between the normals, above which faces are not considered coplanar. The higher this value, the more faces will be considered coplanar and will be simplified. But setting the threshold value is not easy. Reddy in [20] uses the characteristics of the human visual perception to estimate this threshold, but it is most often left to the user and is still a matter of trial and error. While recent algorithms are more flexible than older ones, most are still more appropriate for some types of objects, like those used in medical applications which rarely have sharp edges.

Approximation Error

In order to control the simplification, the approximation error should be measured locally (at each primitive). But for the user to be able to specify the simplification, a global bound should be set on the error. Some algorithms use a local error measure (**A2**, **A3**) or a distance to the original mesh (**A1**, **A5**). Others use a geometric construction to ensure that the simplification does not exceed a certain limit (**A4**, **A6**). One use for the measurement of the error is to determine if a simplification can be used because it's differences with the original model are not noticeable. But only **A4** implements this in a useful way.

Most algorithms allow the user to specify the upper limit for the local approximation error. This is not very intuitive and requires some practicing. Alternatively, some algorithms allow the user to specify how many polygons should be left in the simplification.

Topology Preservation

During the course of the simplification, one choice that the algorithm may have is to simplify the topology. For example, simplifications may lead to filling in a hole or splitting the object into several non-connected parts. While allowing the topology to be modified leaves more room for simplifications, the result is rarely usable since the differences from the original object become too noticeable. **A2** and **A3** allow, but discourage, the simplification of the topology.

Controllable Simplification

It is sometimes interesting to let the amount of simplification vary across the mesh, in order to preserve some parts, yet simplify others more aggressively. This kind of adaptivity involves mainly visual or semantic considerations that may be done *a priori* and are controlled by the user. A few algorithms support this type of adaptable simplification (**A1**, **A2**, **A4**).

Other problems occur when the object's size is big with respect to the scene. In such conditions, some part of the object will be constantly in the foreground and the most detail will thus always be used. The object should be split into several pieces, each of which would have a number of LODs from which to choose. However, there is no universal way of splitting the object and continuity should be maintained between two adjacent pieces so that no errors appear when the two pieces are simplified independently. The simplification algorithm should take care of this.

A1 and **A2** allow the user to insert details in specific parts of the model after the simplification has taken place. Another alternative would be to use a hierarchical structure to describe the various levels of details. To select a LOD would then consist of cutting across the tree, choosing low level detailed nodes for the parts of the object that are close to the observer and higher level, simpler nodes for those that are further away [17].

Polygonal Simplification

Simplification Operators

As stated in [2], a few simple operators (no mathematical formalism is implied here) can be used for removing primitives from a model:

- **Normalization** : removal of degenerated faces or edges and any primitive defined multiple times.
- **Vertex Simplification** : merging of all points included within a volume (either a sphere or a grid cell). Thus, nearby points and faces are combined.
- **Edge Simplification** : removal of all edges shorter than some threshold.
- **Angle-based Simplification** : removal of edges which form a closed angle. Conversely, edges which are aligned are merged.
- **Face size Simplification** : removal of all faces which have an area smaller than some threshold. Holes might have to be filed.
- **Face normal Simplification** : merging of all adjacent faces with near-parallel normals.

These simple operators appear in some commercial packages such as SGI's Cosmo Worlds (see **http://vrml.sgi.com/worlds**). However, they must be used within some kind of control mechanism in order to guide the simplification.

Three Main Categories

Polygonal simplification algorithms fall in three different categories. The figures in this section are inspired from [10].

Geometry Removal

Algorithms in this category produce a simplified version of a model by selecting a number of primitives which must be removed. The selection is made using some kind of heuristic. For example, in **Figure 3**, the algorithm identifies vertices which are in near planar regions. This criterion is relaxed at each iteration until no further vertices can be removed.

Figure 3 - Geometry Removal

This type of simplification is the most popular among recent algorithms. Hinker and Hansen [14] used this technique with a planarity criterion.
Many of the algorithms presented here are in this category, but they vary in their choice of heuristics and their way of measuring the approximation error.

Adaptive Subdivision

This category of algorithms builds an initial simplification which is be to the simplest version of the original model. It then adds more details in the model by subdividing it so that it gets closer to the original model. For example, in **Figure 4**, the initial simplification covers the original model. It is then subdivided and the position of each new vertex is changed to get closer to the original surface.

Figure 4 - Adaptive Subdivision

This type of simplification is not as popular as the previous one because building the initial simplification is not simple in the general case. One simple case is *height fields* and has been studied DeHaemer and Zyda in [7]. Only one of the algorithms presented here belongs to this category.

Sampling

> Finally, algorithms in this category choose a certain number of primitives that should be preserved (as opposed to the first category, which chooses primitives that should be removed). One way of selecting these primitives is by doing a pseudo-random choice, based on some kind of heuristic. Another more powerful method is to sample the model and choose a unique representative for each sample group. For example, while embedding the model in a uniform 3D grid, one vertex can be chosen in each cell to replace all other vertices in its cell.

Figure 5 - Sampling

Rossignac and Borrel [22], weigh each vertex using a function of the local curvature and the length of adjacent edges. Then, using a 3D grid, all vertices in each cell are replaced by that of greatest weight. A new mesh is then generated by removing all edges and faces that have collapsed. This technique is no longer used, mostly because it is difficult to choose the samples in a manner that preserves the overall shape of the object.

Algorithms

We present here six algorithms that were introduced during the last two years. We highlight their main characteristics and their choices with respect to the features presented in the previous sections.

A1 - *Multiresolution Analysis of Arbitrary Meshes* [9]

This is not actually a simplification algorithm, but a preprocessor for another algorithm which produces a **multiresolution representation** of a mesh [8], which is a compact geomorph containing a simple base mesh and a series of wavelet coefficients that are used to introduce details into the mesh. From this representation, a new mesh can use *recursive subdivision* (i.e. where each triangle is subdivided using a 4-to-1 split operator) until the desired amount of detail is reached. Such a mesh is encoded into a multiresolution representation. The algorithm we are presenting here can be used to convert any mesh to one which has the property of recursive subdivision.

This algorithm is an adaptive subdivision algorithm which preserves topology, but does identify any characteristic features in the mesh. Approximation error is measured using the distance to the original mesh. **Harmonic maps** are used at several steps to parameterize a 3D mesh into a planar triangulation. The algorithm has four main steps:

Figure 6 - Four steps in the MRA algorithm
Taken from [9]

1. **Partitioning** : a Voronoi-like diagram is constructed on the original mesh (**Figure 6.1**) using a multi-seed path-finding algorithm in the dual graph of the mesh (where the nodes are the faces of the mesh and the arcs represent adjacency and are weighted using the distance between the centers of adjacent faces). This diagram is then triangulated using a Delaunay-like method and the harmonic maps to straighten the edges (**Figure 6.2**).
2. **Parameterization** : the result is a base mesh (**Figure 6.3**) that is parameterized using a harmonic map. The parametrization is forced to be continuous across the faces so that the number of wavelet coefficients is minimal.
3. **Re-Sampling** : the base mesh is now re-sampled using the 4-to-1 split operator until the mesh is at a certain distance to the original mesh (**Figure 6.4**). Each step is parameterized as in step 3.
4. **Multiresolution Analysis** : the resulting succession of meshes is passed to the multiresolution analysis algorithm to be encoded using wavelets.

This algorithm is appealing in its mathematical formalism. It produces a wide range of simplification, and yet details can be added in specific parts of the mesh. But it is also computationally expensive. Furthermore, extracting a valid mesh from wavelet based representation is expensive.

A2 - *Progressive Meshes* [16]

This geometry removal algorithm also produces geomorphs and is derived from an older algorithm [15]. It searches for planar areas and characteristic edges. The simplification is done by applying an **edge collapse** operator, where an edge collapse produces a new vertex by removing two faces and one vertex. The result is a simplified base mesh and series of **vertex splits** which are an inverse of edge collapses and are used to introduce details into the base mesh. This is called a **Progressive Mesh** and a wide number of simplifications can be extracted from it.

The most important feature of this algorithm is to take into account information such as color, texture, and normal discontinuities over the surface of the mesh. Important features of the model which are represented by this kind of information (and not by simple geometry) are also preserved. For example, **Figure 7** shows how the windows of plane are preserved (take also a look at this GIF Animation to get a better idea of the result).

Figure 7 - Example of a Progressive Mesh
Taken from [16]

The minimization of an energy function is used to guide the simplification. This function has four terms. The first one ensures that the simplified mesh remains close to the original one. The second favors triangles with better proportions. The third term discourages the simplification of color and texture discontinuities. Finally, the last term discourages the simplification of topology and normal discontinuities.

The basic steps of the algorithm are these:

1. Sort the edges using the least cost of simplification. This cost is measured using a variation of the energy function.
2. Apply the edge collapse operator for the edge at the head of the list and record the corresponding vertex split in the progressive mesh structure (including color, texture and normal information).

3. The position of the new vertex is chosen among the two initial vertices and the center of the edge, depending on which one is the closest to the original mesh.
4. Recompute the cost for the edges that have been affected by the operator and reorder the list.
5. If the list is empty or the cost of the next simplification exceeds a certain bound, the algorithm terminates and returns the final progressive mesh. Otherwise, return to step 2.

This algorithm is relatively fast and, because it takes into account the color and the texture, the results are usually very good.

A3 - *Full-Range Approximation of Triangulated Polyhedra* [21]

This geometry removal algorithm preserves planar areas but not the topology. It uses a **region merging** operator, which is roughly equivalent to an edge collapse. This operator is applied if the faces surrounding the edge are coplanar. One of the initial vertices is used as a result of the operator. Approximation error is measured using the distance to the original mesh.

Like the previous algorithm, this one uses an energy function which has two terms. The first one, the **local tessellation error** ensures that the orientation of the normals is preserved and that new faces do not overlap. The second term, the **local geometric error**, keeps the mesh from moving too far from the original mesh. The basic steps of the algorithm are as follows:

1. Sort all edges by increasing cost.
2. Apply the region merging operator to the first edge in the list.
3. Modify the position of the resulting vertex to get it closer to the original mesh.
4. Recompute the cost for modified edges and sort the list. The cost is accumulated at each iteration so that the simplification is more evenly distributed across the mesh.
5. If the list is empty or if the cost for the next edge is higher than a threshold, the algorithm terminates. Else return to step 2.

Although the basic assumptions are quite different, one can notice that this algorithm is very similar to the previous one. It is also a lot more efficient at preserving the features of the objects than was its ancestor [22].

A4 - *Simplification Envelopes* [6]

This geometry removal algorithm was initially conceived by Varshney [23]. It preserves planar areas and sharp edges, as well as topology. The main goal of this algorithm is to use no error measure but only a geometric construction to control the simplification. **Simplification Envelopes** are two surfaces constructed on each side of the original surface using a user specified offset and making sure these surfaces do not self-intersect. The space between the two surfaces is then used to build a new surface, the only constraint then being that the new polygons should not intersect with either surface. This reconstruction can be done in several ways, of which we will here present only one.

The amount of simplification is controlled by the offset used for constructing the surfaces. The case where envelope surfaces are most likely to self-intersect is along sharp edges of the original mesh, where there will not be much room to build one of the surfaces. The surfaces which self-intersect must then be moved closer to the original mesh until the condition is corrected. Thus, near sharp edges the spaces between the two surfaces will be smaller and fewer simplifications will be permitted. Conversely, in planar areas, the distance will be maximal, and so will be the simplification.

Figure 8 - Building inner and outer envelopes for a triangle
Taken from [6]

The algorithm starts by constructing the envelopes (**Figure 8**) :

1. Offset the outer surface along the normals to the vertices by a fraction of the desired final offset.
2. If, for any vertex, the surface self-intersects, cancel the move for that vertex.
3. Repeat 1 and 2 until either no further increment can be made without intersection, or the offset has reached the desired value.
4. Repeat 1 to 3 for the inner surface.
5. Repeat 1 to 3 for the border tubes. These are built along the borders of non-closed objects to allow for simplification there also.

This iterative manner of building the offset surfaces produces non-optimal results, i.e. the envelopes are sometimes closer to the original mesh than they should be. Computing the optimal solution requires evaluating Voronoi edges in the 3D space, which is computationally very expensive.

The algorithm then goes on to generating the simplified mesh. For each vertex of the initial mesh:

1. Remove the vertex and the adjacent faces.
2. If possible, iteratively fill the hole by triangulation using the biggest faces possible and ensuring they do not intersect with the offset surfaces. If not possible, cancel the removal and try the next vertex.

This algorithm is appealing because it does not use any measure of the error. The envelopes are the only control over the simplification. It is computationally expensive though, especially during the envelope construction phase.

A5 - *Surface Simplification Inside a Tolerance Volume* [12]

This geometry removal algorithm is somewhat combination of **A3** and of a reversed version of **A4**. It removes edges as in **A3**, but the control is done using **Tolerance Volume**. But, where as in **A4** the volume was built around the original surface, in this algorithm it is built around the simplified surface, and the simplified is constrained not to let the original get out of the tolerance volume. The amount of simplification is controlled by the thickness of that volume.

The tolerance volume is computed as a sphere around each vertex, and represents the accumulation of the error introduced by simplification which lead to the creation of the vertex (by applying the edge collapse operator). It is then interpolated across edges and faces (**Figure 9.1 and 9.2** respectively). One of the nice properties of the algorithm is to ensure that the volume of the object is not changed by the simplification (within a given tolerance). This is very useful for some application domains such as medical data visualization.

Figure 9 - Building and maintaining the tolerance volume

Taken from [12]

The basic steps of the algorithm are the following. The initial simplification is simply a copy of the original mesh. The error volumes are set to null. For each edge by order of increasing length:

1. Apply the edge collapse operator. The position of the new vertex is computed by resolving equations that ensure it remains close to the initial mesh and that the volume is preserved.
2. Check that normals of modified faces have not been reversed. If they have, cancel the edge collapse and go to the next edge.
3. Check that new faces have good proportions. This is evaluated using a function of the surface and the length of the edges and checking that this function does not change too much after the operator is applied. If it has, cancel the edge collapse and go to the next edge.
4. Update the edge list to take into account the modifications introduced by the simplification.
5. Compute the new tolerance volume so that it contains the previous volume (**Figure 9.3**).
6. If the diameter of the tolerance volume for the new vertex is greater than a given threshold (i.e., that the original mesh is not enclosed within the tolerance volume), then the edges that share this vertex are removed from the list and no further simplification will take place in that area.

A6 - *Mesh Simplification* [1]

This geometry removal algorithm is original in that it does not measure the approximation error, but uses a clustering process to ensure that simplification is restricted to certain areas. This clustering is done by searching for **characteristic edges** and planar areas. Characteristic edges are those that are shared by faces which form an angle greater that some given threshold. Typically, there are many characteristic edges in areas where there are few coplanar faces.

Figure 10 - Steps in the Mesh Simplification Algorithm

Inspired by [1]

The algorithm does the following operations:

1. Look for characteristic edges and label the vertices using the number of characteristic edges sharing it. A ``0'' vertex has no characteristic edges leaving from it.
2. Build clusters of coplanar polygons. Look for edges between two ``0'' vertices and use them to build pairs of coplanar faces where the edges between two ``0'' vertices are not connected (**Figure 10.1**).
3. Within each cluster, apply an edge collapse operator until no further simplifications can be made (**Figure 10.2 to 10.4**).
4. Look for collinear characteristic edges leaving from a ``2'' vertex, such as the vertices on the edges of a cube (**Figure 4**). This can not be simplified by an edge collapse operator since the two adjacent faces are not coplanar. These edges are merged and the resulting vertex is pushed towards the other vertex. The result is to move the vertices on a characteristic edge towards its extremities (**Figure 10.5**).
5. Look for ``0'' vertices surrounded by only non-``0'' vertices, like the one in the center of one side of a cube (**Figure 10.5**). Such vertices cannot be removed by the previous method because there are no other adjacent ``0'' vertices to build a cluster with. So such vertices are removed, and the hole is triangulated (**Figure 10.6**).

This algorithm is attractive by its simplicity, but the simplification is not always optimal. Better results can be achieved by gradually increasing the planarity and collinearity thresholds while the algorithm proceeds.

Comments and Perspective

It should be obvious from the previous section that, although most algorithms have a unique way of controlling and applying the simplification, the basic process is always the same: look for planar areas, simplify them, look for sharp edges, preserve them. However, other distinctions can be made and further enhancements should be studied.

- Some algorithms control the simplification using a energy function (or a distance) and try to optimize it (**A1**, **A2**). Others use the local geometry and measure things such as curvature (**A3**, **A4**, **A5**, **A6**).

- Some algorithms minimize the approximation error whereas others minimize the number of triangles.
- Most algorithms measure a local approximation error when removing a primitive (**A2**, **A3**, **A4**, **A5**) whereas others measure a global error (**A1**). Some rare algorithms don't measure any approximation error (**A6**).

Further Enhancements

More advanced simplification might be performed if the algorithms considered information other than the geometry.

Colors, Textures and other Information

Of the algorithms presented here, **A2** is the only one to use information such as color and texture. But as models gets richer, this kind of data will be more and more vital to the user. They make the object more identifiable and thus should be preserved. Discontinuities are most important, like the change in color for the windows of the plane in **Figure 7**. This simplification should be done using the characteristics of the human visual perception.

Topological Information

Most algorithms we saw try to extract topological information from the geometry, such as where sharp edges might be. However, because such information is not present in the model, it cannot be simplified as such. For example, if the algorithm had the knowledge that a sharp edge was in fact a topological circle, it might be able to change into something simpler, like a pentagon.

In addition, changes in color and textures should be treated in much the same way as topological discontinuities in the geometry. A LOD creation algorithm based on a topological model unifying geometry, color and texture should be investigated.

Relationship to other Models Based on Non-Polygonal Primitives

While most modern real-time virtual reality hardware uses polygonal primitives, the models are commonly produced using CAD packages which manipulate higher level primitives such as parametric surfaces. In this context, [3] presents a model for adaptively choosing an assembly of parametric patches, by smoothing for a given point of view, producing a simplified polygonal approximation of the surface. Furthermore, these models contain topological information which might be used. Thus, it would interesting to conceive a model for the integration of surface-based and volume-based models as well as polygonal models within the same simplification algorithm.

Conclusion

New polygonal simplification algorithms are now capable of producing satisfactory levels of details with respect to visual and geometric requirements. However, no matter which algorithm is used, much practice is still required before being able to predict the amount of simplification and specify correct values for the various parameters. Moreover, no algorithms use the characteristics of the human visual perception to set these parameters. LOD generation very much remains a modeling activity.

On the other hand, the complexity of the objects involved in modern virtual reality simulations increases every day. Texture and lighting information (such as that produced by radiosity computations) is added to produce more realistic rendering. Thus algorithms should evolve to take into account this information and adapt them during the simplification process.

Finally, the creation and selection of LODs should be integrated to scene management techniques [5]. Scene partitioning and visibility relationship computation may help the simplification in the choice of the amount of simplification that is required for a particular scene.

Additional reading may be found on the author's Levels of Detail web page at **http://www.limsi.fr/Individu/krus/LODS/**.

References

1

Algorri, M.-E. and Schmitt, F. Mesh simplification. 1996. In *Proceedings of EuroGraphics'96, Computer Graphics Forum* (Futuroscope, Poitiers, France), volume 15, pages 77-86.

2

Astheimer, P. and Pöche, M.-L. Level-of-detail generation and its applications in virtual reality. 1994. In *Virtual Reality Software and Technology* (Proceedings of VRST'94, August 23-26, 1994, Singapore), pages 299-312.

3

Bourdot, P. *Gestion de Scènes Complexes en Amont des Algorithmes de Rendu*. PhD thesis, Université de Droit, d'Economie et des Sciences d'Aix-Marseille III.

4

Chazelle, B., Dobkin, D. P., Shouraboura, N., and Tal, A. 1995. Strategies for polyhedral surface decomposition: An experimental study. In *Proceedings of the 11th Annual ACM Symposium on Computational Geometry*, Vancouver, British Columbia, Canada, June 5-7.

5

Clark, J. H. 1976. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19:547-554.

6

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. 1996. Simplification Envelopes. In *Computer Graphics (SIGGRAPH '96 Proceedings)*.

7

DeHaemer Jr., M. and Zyda, M. 1991. **Simplification of objects rendered by Polygonal Approximations**. *Computers & Graphics*, 15(2):175-184.

8

DeRose, T. D., Lounsbery, M., and Warren, J. 1993. ***Multiresolution analysis for surface of arbitrary topological type***. Technical Report 93-10-05, Department of Computer Science, University of Washington, Seattle, WA.

9

Eck, M., DeRose, T., Duchamp, T., Hoppe, H., Lounsbery, M., and Stuetzle, W. 1995. **Multiresolution Analysis of Arbitrary Meshes**. In *SIGGRAPH '95*. Also as TR95-01-02, Department of Computer Science and Engineering, University of Washington.

10

Erickson, C. 1996. ***Polygonal Simplification : An Overview***. Technical Report TR96-016, Department of Computer Science, UNC-Chapel Hill, January.

11

Funkhouser, T. A., Sequin, C. H., and Teller, S. J. 1992. Management of large amounts of data in interactive building walkthroughs. In Zeltzer, D., editor, *Computer Graphics* (1992 Symposium on Interactive 3D Graphics), volume 25, pages 11-20.

12

Gueziec, A. ***Surface simplification inside a tolerance volume.*** Technical Report RC 20440, IBM Research Division, T. J. Watson Research Center.

13

Heckbert, P. and Garland, M. 1994. **Multiresolution modeling for fast rendering**. In *Proceedings of Graphics Interface '94*. (Banff, Alberta, Canada), Canadian Information Processing Society, pages 43-50.

14

Hinker, P. and Hansen, C. 1993. **Geometric optimization**. In *Visualization'93*, pages 189-195.

15

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1993. **Mesh optimization**. In Kajiya, J. T., editor, *Computer Graphics* (SIGGRAPH '93 Proceedings), volume 27, pages 19-26.

16

Hoppe, H. 1996. **Progressive meshes**. In *Computer Graphics* (SIGGRAPH'96 Proceedings).

17

Luebke, D. 1996. ***Hierarchical structures for dynamic polygonal simplification***. Technical Report TR 96-006, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina.

18

Maciel, P. W. C. and Shirley, P. 1995. **Visual navigation of large environments using textured clusters**. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics*.

19

Reddy, M. 1995. **A survey of level of detail support in current virtual reality solutions**. In *Virtual Reality: Research, Development and Applications*, 1 (2).

20

Reddy, M. 1996. **SCROOGE: Perceptually-Driven Polygon Reduction**. In *Computer Graphics Forum*, 15(4):191-203.

21

Ronfard, R. and Rossignac, J. 1996. *Full-range approximation of triangulated polyhedra*. Technical Report RC 20423, IBM Research Division, T. J. Watson Research Center. Also to appear in Eurographics '96.

22

Rossignac, J. R. and Borrel, P. 1992. Multi-resolution 3D approximations for rendering complex scenes. In Falcidieno, B. and Kunii, T. L., editors, *Geometric Modeling in Computer Graphics*, pages 455-465, Genova, Italy. Springer-Verlag.

Also published as technical report RC 17697 (#77951), IBM Research Division, T. J. Watson Research Center.

23

Varshney, A. 1994. [***Hierarchical Geometric Approximations***]. Ph.D. thesis, Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175. Also available as TR-050-1994.

24

Véron, P. and Léon, J.-C. 1996. Synthèse et Analyse des différentes approches de simplification de polyèdres. In *Congrès Numérisation 3D : Design et Digitalisation, Création Industrielle et Artistique.*

25

Watson, B., Walker, N., Hodges, L., and Worden, A. 1996. ***Effectiveness of peripheral level of detail degradation when used with head-mounted displays.*** Technical Report 96-04, Graphics, Visualization & Usability (GVU) Center, Georgia Institute of Technology.

Authors

Mike Krus is a PhD student of the Univerisity Paris XI, Orsay, France. His work was initiated by the **LIMSI/CNRS**, a public research laboratory, and **Electricité de France** , a public electrical utility.
Contact: EdF/DER - IMA/TIEM/CAO, 1, av du Gle de Gaulle, 92141 Clamart Cedex.
Tel: +33 1 47 65 42 73. Fax: +33 1 47 65 34 24.
Email: **Michael.Krus@der.edfgdf.fr** .

Patrick Bourdot is a researcher in Computer Graphics at **LIMSI** (Paris XI University - Orsay), a laboratory of the French **National Center for Scientific Research (CNRS)**. After a diploma of Architect (1986), he obtained in 1992 a Ph. D. in Computer Science at the University of Aix-Marseille III. His main interests are levels of detail on parametric surfaces by a hierarchical and multiple sub-definition approach, 3D reconstruction with parametric curves and surfaces from image analysis, knowledge representation to manage "reactive" objects in 3d modelling, multimodal user interfaces with vocal interaction and graphic recognition for CAD systems.
Contact: LIMSI/CNRS, BP 133, 91403 Orsay Cedex.
Tel: +33 1 69 85 81 21. Fax: +33 1 69 85 80 88.
Email: **pb@limsi.fr**.

Françoise Guisnel is a researcher in virtual reality at the CAD group of **Electricité de France**'s **Research and Development Division**.
Contact: EdF/DER - IMA/TIEM/CAO, 1, av du Gle de Gaulle, 92141 Clamart Cedex.

Guillaume Thibault is a researcher in virtual reality at the CAD group of **Electricité de France**'s **Research and Development Division**.
Contact: EdF/DER - IMA/TIEM/CAO, 1, av du Gle de Gaulle, 92141 Clamart Cedex.