Texture dependent refinement of multiresolution meshes

Andreas G. Schilling, Reinhard Klein

ISSN 0946-3852
WSI-98-2

Wilhelm-Schickard-Institut für Informatik
Graphisch-Interaktive Systeme
Auf der Morgenstelle 10/C9
D-72076 Tübingen
Tel.: +49 7071 29-75462
Fax: +49 7071 29-5466

email: {andreas|reinhard}@gris.uni-tuebingen.de
URL: http://www.gris.uni-tuebingen.de/

Texture dependent refinement of multiresolution meshes

Andreas Schilling and Reinhard Klein *

Abstract

State of the art multiresolution modeling allows to selectively refine a coarse mesh of an object on the visually important parts. In this way it is possible to render the geometry of a given object accurately with a minimum number of triangles. The criteria used in current approaches take care of geometric error and even of shading errors. If however texture mapping is used it is inevitable to control displacements and distortions of the texture during refinements.

In this paper we describe a new approach for better estimation of the errors due to displaced and distorted texture coordinates in simplified meshes. This allows for accurate rendering of textured scenes with a minimum number of triangles.

1 Introduction

Recently multiresolution modeling for simplicial geometric models (any dimension, non-orientable, non-manifold, non-regular) have gained massive interest. Multiresolution models (MRMs) provide the basis to handle, visualize and transmit over the network and edit very large data sets [9, 2, 1, 3, 4, 7, 8, 10]. These models provide the basis for the geometric approximation of an object with different levels of detail. As a growing number of applications rely not only on the geometry of the objects but make also extensive use of textures, the incorporation of textures into the multiresolution models becomes a new challenge. Interestingly, this problem has not been addressed by the numerous publications about mesh simplification, multiresolution modeling and view-dependent selective refinement so far.

1.1 The Texture Problem

In the process of texture mapping, 2D-images are mapped onto surfaces in 3D. Normally the surface is triangulated and texture coordinates are assigned to each vertex. In this way triangular parts of the image are linearly mapped onto the triangles of the model. Unfortunately arbitrary surfaces are not developable and therefore, the mapping of 2D-textures onto the surface is not isometric, this means that the associated affine transformation is not the same for all triangles of the surface. If during the simplification process several triangles with different affine mappings are replaced by one triangle (with only one affine mapping) errors a inevitable, see Figure 1. Only if in addition to the geometric errors also the distortion of the texture is measured and controlled, texture mapping on simplified surfaces delivers reasonable results.

1.2 Outline of the solution

A measure for the texture distortion in the simplified mesh is needed. The solution for that is to use the 3D-distance between points in the original mesh and the corresponding points in the simplified mesh (the ones that get the same texture coordinates) instead of the geometric distance like the Hausdorff- distance between the two meshes [4]. This measure exactly describes how far a textured point is moved from its original location when the simplified mesh is used. This measurement has to be done during the generation of the multiresolution model. Later on at rendering time our measure allows us to selectively refine the model until the displacement and distortion of the texture on the simplified model as well as the geometric error between simplified and original model is less than a predefined bound in screen space, e.g. half a pixel.

In addition the necessity of refinement depends on the contents of the texture: for parts of the texture where the color doesn't change of course only the geometric error matters.

In the reminder of this paper we first briefly review the principles of multiresolution modeling. Then we discuss the new error measure and derive a possible texture-enhanced multiresolution model.

* Universität Tübingen, Auf der Morgenstelle 10 / C9, 72076 Tübingen, Germany.
E-mail:{andreas|reinhard}@gris.uni-tuebingen.de
http://www.gris.uni-tuebingen.de

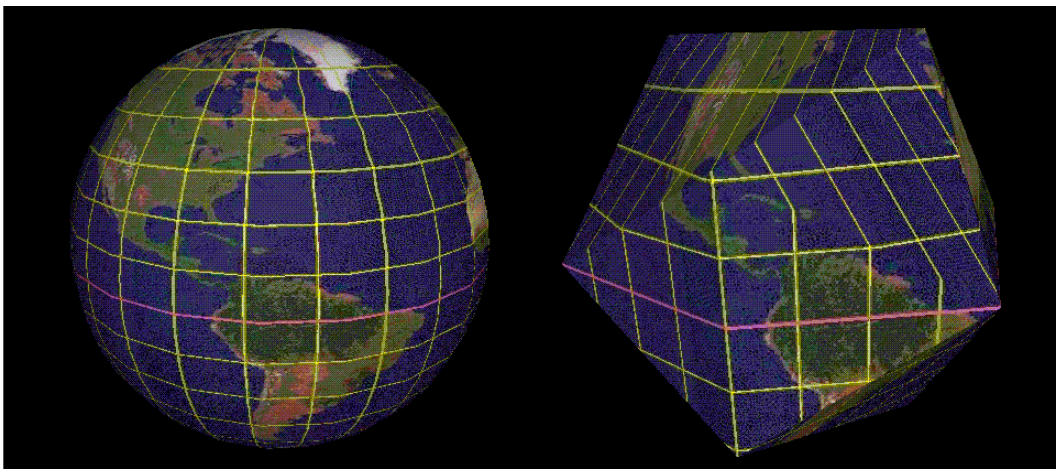

Figure 1. The icosahedron on the right side is still a good geometric approximation for the globe. The geometric approximation error is about ± 350 miles, but the texture distortion gets close to 2000 miles. Close to the equator the approximation of the texture is still relatively good, but approaching the poles the approximation errors become larger and larger. (Distance between two meridians and between two parallels of latitude is 15 degrees.)

2 Review of multiresolution models

2.1 Generating the multiresolution model

The generation of a MRM of an object generally involves a sequence of local simplification operations like vertex removal, edge collapse, triangle collapse or vertex clustering. The sequence of local simplification operations defines a sequence of coarser and coarser approximations of the original model, the MRM. How this sequence is generated depends on the various simplification algorithms. In general a mesh simplification algorithm starts with the finest triangulation in 3D space approximating the original model. Then it simplifies the starting triangulation by clustering vertices, by collapsing edges or triangles or by removing vertices from the current triangulation and retriangulating the resulting holes. This is done until no further simplification step can be performed. In many algorithms the order in which the simplification steps are performed is determined by a priority queue. A cost function is evaluated for each possible simplification operation and the one with the lowest cost is performed. In general the cost function represents the error (geometric distance) between original and simplified mesh.

2.2 Selective refinement of multiresolution models

If the inverse local simplification operations are known (e.g. vertex split as the inverse of edge collapse operation), we are able to refine a coarse approximation of the model by reversing the whole simplification process. However, if we want to perform only selective refinement we have to find a way to skip parts of the inverse simplification process and thereby change the sequence of refinement operations. Of course this is not arbitrarily possible (e.g. we cannot split a vertex which is not present in the current mesh). The dependencies between the different simplification steps define a hierarchy that can be described by a directed acyclic graph of modification operations or the associated triangles. Therefore, a general selective refinement algorithm starts with a crude approximation of the model and checks for each triangle if refinement is needed. If yes, the algorithm has to take care that all predecessor operations of the needed refinement operation have already been performed. The next section describes the measure that can be used to decide about the need of further refinement of a certain triangle.

3 Texture Distortion and Geometric Errors

3.1 Point to point correspondence and distortion

For the calculation of the distance between corresponding points we assume that texture coordinates (s,t) are assigned to every vertex in the original mesh. Both the original and simplified model are rendered with linear texture coordinate interpolation on the triangles. In order to compute the error between the triangle in the simplified model and the corresponding area in the original mesh the following technique is used. The triangulation of the original model

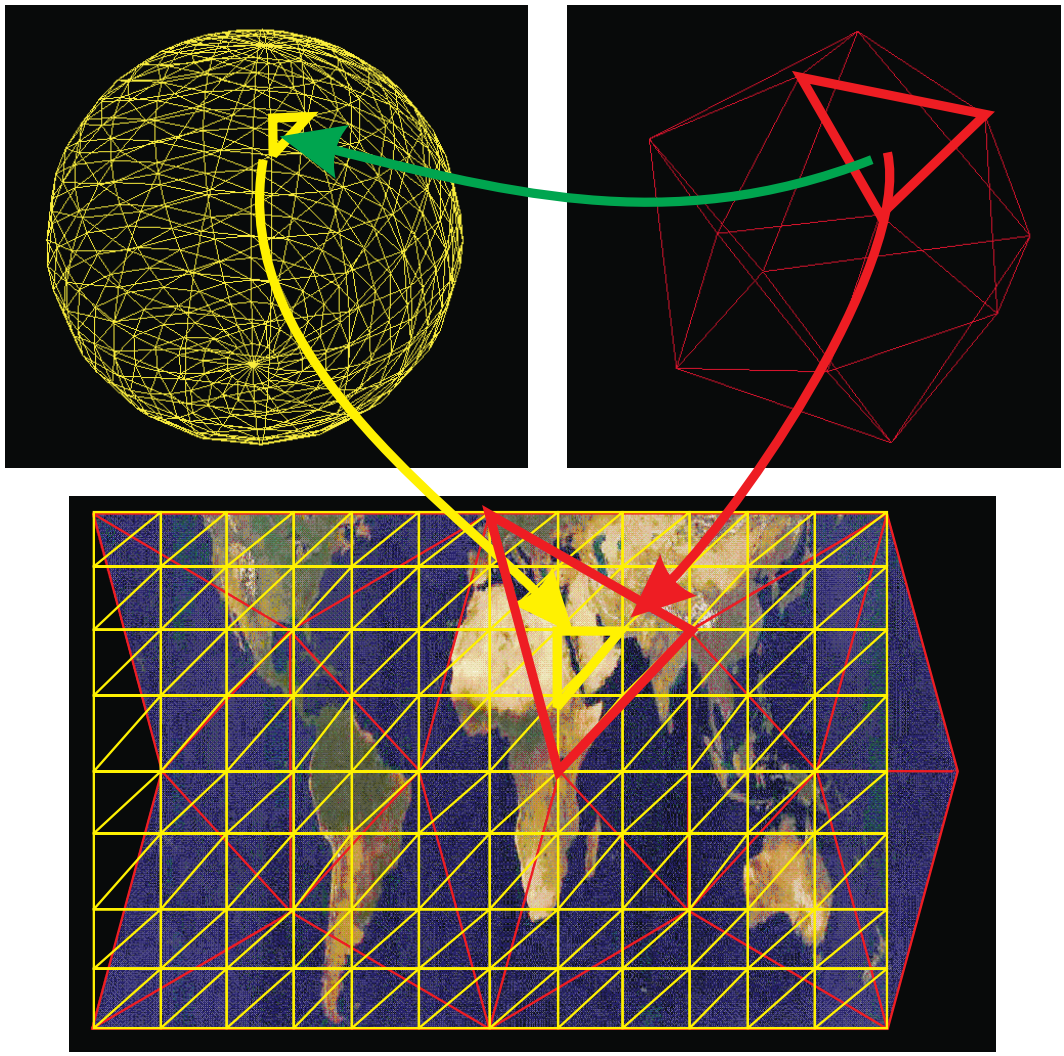

Figure 2. Mapping of original and simplified triangulations into texture space. By these two mappings a point-to-point relation between points on the original mesh and points on the simplified mesh can be established. This relation exactly describes the distortion of the texture in the simplified model and is therefore used to define the error measure.

defines a corresponding triangulation in texture space, see Figure 2. Every point in the texture space corresponds to two points, one on the simplified and one on the original mesh. The interesting error is the geometric distance vector between those two points. The maximum length of all these distance vectors bounds the geometric error between the simplified and original model. Therefore, it can be used during the generation of the MRM as sorting criterion for the priority queue. Secondly, at rendering time this error can be directly converted into a screen space error.

3.2 Computation of the errors

To calculate the maximum length of all distance vectors it is sufficient to calculate it for each triangle Δ_{simp} of the simplified triangulation

$$\epsilon_{\Delta_{simp}} = \max_{(s,t)\in\Delta'_{simp}} ||M_{simp}(s,t) - M_{orig}(s,t)||,$$

where M_{simp} and M_{orig} denote the mapping from texture coordinates onto the simplified and the oringinal meshes, respectively[1]. Note that this maximum error can only occur on the vertices and edge- intersections of the triangles Δ'_{orig} and Δ'_{simp}. This leads to the following algorithm: First, all vertices and all edges in texture space of the original triangulation that intersect with Δ'_{simp} are collected. With neighborhood information of the original mesh this can easily be done using an active edge list algorithm. In the next step the intersection points between the collected edges and the edges of Δ'_{simp} are computed. In the last step the corresponding points of the inner vertices and intersection points are determined using M_{simp} and M_{orig}. For this purpose, the barycentric coordinates of these points in texture space are used.

4 The texture enhanced multiresolution model

In a simple texture enhanced multiresolution model it is sufficient to store for each triangle the distance ϵ_Δ that occurred when in the process of simplification this triangle was generated. Using this information at extraction time we can immediately decide if a given triangle of the simplified triangulation is valid or if further refinement is needed. The only criterion that has to be checked is that spheres with radius ϵ_Δ at the vertices of the triangle project to the screen smaller than the allowed screen- space error of e.g. half a pixel. In this way, the existing MRMs can easily be modified to do correct rendering of textured models. It is surprising that ϵ_Δ which is known as *parametric distance* (with the texture space as parameter space) to our knowledge was not used until now for this purpose of texture dependent simplification.

[1]Triangles in the texture space are denoted with dashes.

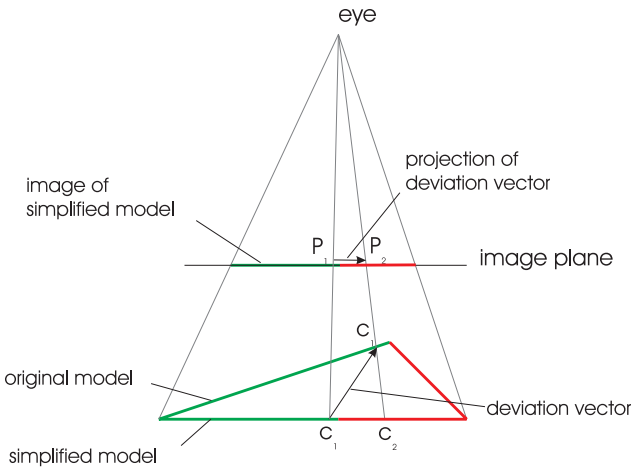

Figure 3. The color deviation caused by the use of the simplified model instead of the original model. The deviation vector tells that the color c_1 at screen location P_1 should be visible at screen location P_2, where the wrong color c_2 appears. Note that the correct position of color c_1 in the image space depends on the viewing position.

4.1 Improving the model

However, this simple model does not take into account the texture itself. Homogeneous parts of the texture are treated in the same way as detailed parts with high contrast. For changing textures (e.g. video-textures) this approach makes sense. But in the common case of static textures we can do better. The key observation is that even a large deviation of texture coordinates cannot be recognized if the texture color doesn't change in the regarded area or at least doesn't change in the direction of the deviation.

How can we find out, if there is a difference between the rendered image of the simplified and the rendered image of the original model? To answer this question let us project the simplified model onto the screen and assume for the moment that for each point of the simplified model the three-dimensional deviation vector defined by the point-to-point relation above is available. The projections of these deviation vectors onto the screen tell us how far and where the points would move if we would use the original instead of the simplified model, see Figure 3. Of course, for a fast decision where to selectively refine the model at rendering time we need a very simple measure for the expected color deviations. Unfortunately, it is impossible to calculate color

changes independent from the viewing position. Therefore, it is impossible to store the expected color deviations already in the MRM. Instead, a measure is needed that enables us to calculate these deviations at rendering time. In order to be fast, this measure must deliver a single quantity for each triangle of the simplified model.

4.2 Measuring color deviations

A possible measure could be calculated from a three-dimensional bounding volume for the geometric deviation vectors over a triangle and a two-dimensional bounding volume of the color gradients in the texture space. At rendering time from the projections of both of these bounding volumes onto screen space a worst-case color deviation could then be calculated. This approach has two disadvantages: First the costs in memory and computation time to store and combine the two bounding volumes in screen space are too high. Secondly, there might be situations where the combination of color gradient and geometric deviation vector in each point would lead to small color changes, but as bounding volumes over a triangle are used we only get a worst case estimation, where the maximum color gradients over a triangle are combined with the maximum deviation vectors over a triangle, which in their turn might correspond to totally different locations on the triangle.

A better measure can be developed using the following observation: if the geometric deviation vectors are decomposed in a tangential and a normal component (with respect to the simplified triangle), the color changes caused by the tangential component do not depend any more on the viewing direction and can thus be calculated in advance during the generation of the MRM. In this case instead of using the worst-case combination of deviation vector and color gradient for the whole triangle, the exact resulting color-deviations at each point are used. We get these deviations basically by subtracting an appropriately warped version of the texture from the original texture. To get an estimation for the total color deviation, the color deviation caused by the normal component of the deviation vector must then simply be added. A fast simple bound for this color deviation can be calculated by multiplying the length of the screen projection of the normal component with the maximum projected color gradient over the region of the triangle, enlarged by the possible projection of the normal component. Since the length of this projection is in general unbounded (at small viewing angles with respect to the triangle) this measure makes only sense if the viewing angle is not smaller than a predefined minimum. This can easily be checked at rendering time. As the projection is unknown at simplification time, neither the projected color gradient nor the length of the projected normal component are known in advance. Therefore, we store only the product of the length of the normal component and the maximum absolute value of the color gradient in the regarded area. Later at rendering time the actual projection is taken into account.

4.3 The Mip-Map-levels

The above error measure allows high geometric deviations even at small viewing distances if the texture permits it (small color changes). During rendering time the distance from which a certain triangle may be seen is a priory unknown and therefore, the Mip-Map level used to render the triangle is also unknown. The problem here is, that different Mip-Map-levels contain different color differences and therefore, the same geometric error can lead to different color deviations. As an example, imagine the sand in the desert. Unless we get quite close, the sand appears to have a single color in this case texture distortion cannot be recognized. The situation changes abruptly when we get close enough and the Mip-Map-level changes so that the single grains of the sand can be seen. Therefore, the error measures defined above are dependent from the Mip-Map-level and must be stored for each Mip-Map-level independently.

5 The extraction algorithm

The goal of the multiresolution model is to minimize the number of polygons that are sent to the rendering pipeline, without affecting the quality of the resulting image. To achieve this, the multiresolution model allows the extraction of selectively refined models either by coarsening or refining the current triangulation. The decision if a) a given triangle may remain in the triangulation or if b) its surrounding area must be refined further or if c) a coarser triangulation of its surrounding area is possible, is the most important step in this process. For an implementation this can be reduced to the single question if a given triangle fulfills the quality requirements or not [6]. To answer this question we have to investigate the characteristics of that part of the original surface that corresponds to the triangle in question. In the following we call this part of the original surface *corresponding surface* for simplicity. The flow diagram of the algorithm we use for this decision is shown in Figure 4.

5.1 Handling backfacing surface parts

The first question in the algorithm is to decide if a triangle $\Delta(v_{i_1}, v_{i_2}, v_{i_3})$ and its corresponding part on the original surface is on the backside with respect to the current viewing position. This first step has been described by several authors ([11, 4, 5] to mention only a few).

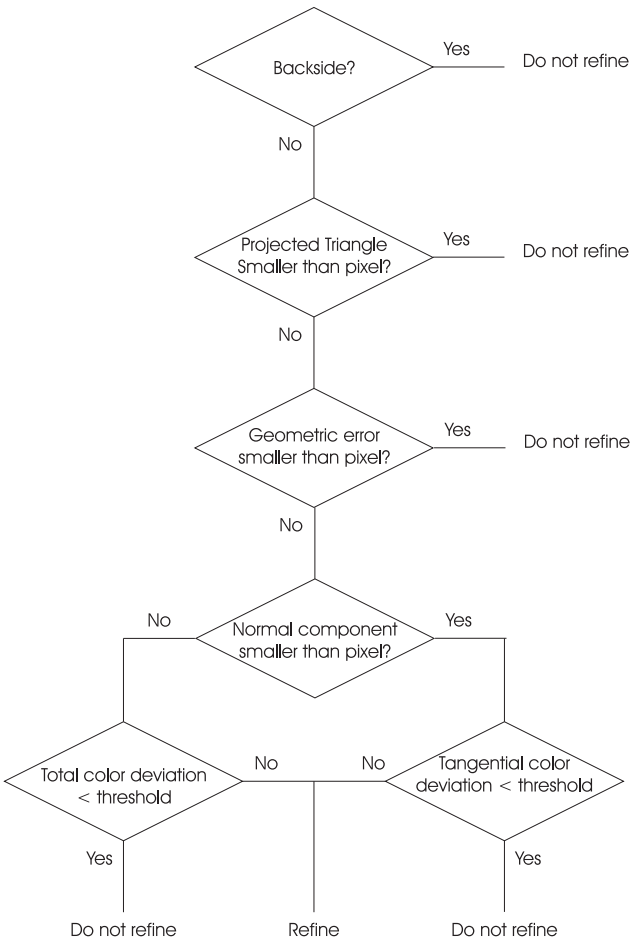

Figure 4. Flow diagram of the extraction algorithm.

5.2 Controlling the geometric error

To control the geometric error, the parametric distance between original and simplified triangulation must be converted to a screen space error. Several approaches to this problem have been published and can also be used to convert the parametric distance [11, 4, 5]. If the projected error is less than half a pixel, of course no further refinement is needed. A further significant reduction would be possible by avoiding the refinement outside of the view frustum. However, to support fast changes of the viewing direction in VR-applications refinement outside the view frustum is inevitable.

5.3 Controlling the color deviation

In contrast to previous work, we can avoid refinement even in cases where the projected screen space error is larger than half a pixel, if in the result the color doesn't change. To check the color changes we consider the precalculated contribution from the tangential component for the triangle in question and add the contribution resulting from the geometrical error in normal direction multiplied by the maximum color gradient. Because of the projection, the second contribution is more complicated. Therefore, we first check if the normal component projects to a screen length smaller than half a pixel, in which case it can be neglected and only the stored tangential contribution has to be checked (right branch in the flow diagram). In the other case we cannot neglect the normal contribution. As described above, if the viewing angle with respect to the triangle is smaller than a predefined threshold ($45°$ are reasonable), we refine. In the other case we use the stored product of the length of the normal component of the parametric error and the maximum color gradient and multiply it with the cotangent of the viewing angle with respect to the triangle, see Figure 5. After addition of the stored color deviation caused by the tangential component, we get a reasonable bound for the total color deviation.

6 Conclusion

This paper deals with the new and interesting problem of combining multiresolution modeling and texture mapping. A texture enhanced multiresolution model is presented, that can be used to detect and avoid texture distortions. An improved version of the model takes the contents of the texture into account and allows to achieve comparable results with coarser geometry. The presented results can be applied to various existing types of multiresolution models.

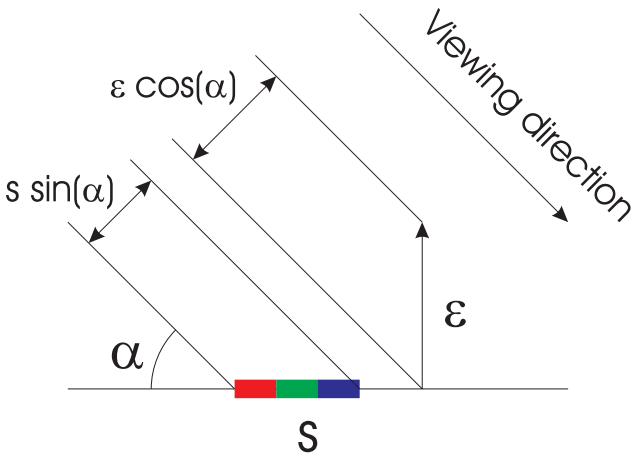

Figure 5. The product of the length of the normal component and the color gradient has to be multiplied with cotan(α) to correct for the screen projection, as the normal component ϵ gets shortened with $cos(\alpha)$, and the color gradient has to be multiplied with $1/sin(\alpha)$.

References

[1] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. Technical Report CNUCE: C96-021, Istituto per l'Elaborazione dell'Informazione - Condsiglio Nazionale delle Richere, Pisa, ITALY, July 1996.

[2] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. P. Brooks, Jr., and W. Wright. Simplification envelopes. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, Aug. 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[3] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[4] H. Hoppe. View-dependent refinement of progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1997 (ACM SIGGRAPH '97 Proceedings)*, pages 189–198, 1997.

[5] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 199–208. ACM SIGGRAPH, Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.

[6] E. Puppo. Variable resolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

[7] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, Sept. 1996.

[8] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scences. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer Verlag, 1993.

[9] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[10] G. Turk. Re-tiling polygonal surfaces. In E. E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[11] J. C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In H. Rushmeier, editor, *Visualization '96 Proceedings*, volume 30(4), Aug. 1996.