C·A·D·P·I·P·E

SIXTH FRAMEWORK PROGRAMME HORIZONTAL RESEARCH ACTIVITIES INVOLVING SMES CO-OPERATIVE RESEARCH

Contract for: CO-OPERATIVE RESEARCH PROJECT

Project acronym: CADPIPE
Project full title: Cad Production Pipeline
Proposal no.: 512897

Contract no.: COOP-CT-2004-512897

State-of-the Art Hardware Based Rendering, Parametric Surfaces and Data Reduction

Version 1.0

21.6.2005

Aydin Ozturk

Version history

Version	Date	Author(s)	Reviewer	Description
0.1-1	31.05.2005	Aydin Ozturk		First draft
0.1-2	31.05.2005	HyungSeok Kim, Stephane Garchery, Nadia Magnenat-Thalmann		Modification on the some style issue and clarification on the contribution.
1.0	21.06.2005	HyungSeok Kim, Stephane Garchery, Nadia Magnenat-Thalmann		Conclusion updated

Contact information

Aydin Ozturk
Ege Universitesi
Uluslararası Bilgisayar Enstitüsü
Bornova, 35100 İzmir
Tel. +90 232 3887228/110, fax +90 232 3887230
Email: ozturk@ube.ege.edu.tr
Web: http://www.ube.ege.edu.tr/~ozturk

Contents

1. Hardware Based Rendering

Aydin Ozturk, Ahmet Bilgili, Serkan Ensoner
International Computer Institute, Ege University, May 2005
www.ube.ege.edu.tr
CADPIPE Internal Restricted Usage

Abstract

The objective of this report is to provide a survey of relevant topics and focus on the state-of-the art techniques on hardware based rendering. Lighting models based on Bidirectional Reflectance Distribution Functions were considered and algorithms and corresponding hardware implementations were reviewed.

Keywords: *real-time rendering, lighting and shading, graphics hardware, BRDFs*

1.1 Introduction

1.1.1 Purpose of this report

This report is a technical document of the research project CADPIPE (CAD Pipeline) on Hardware *Based Rendering*. The objective of this report is to provide a survey of relevant topics and focus on the state-of-the art techniques.

1.1.2 Background

In computer graphics one usually starts with an application program and ends with an image. This process can be considered as a black box. Within each black box, a number of tasks including geometric transformations, clipping, shading, hidden surface removal, occlusion culling, and rasterization are carried out. Whatever the strategy is used to organize these tasks, every geometric object must pass through the system and a color must be assigned to every pixel in the buffer that is displayed [1].
Today, most of the graphics systems employ rendering techniques based on pipeline architecture. There are three major stages in a real-time *rendering pipeline* namely *application, geometry, and rasterization* [2]. A typical result of the application stage is

sets of vertices that define a group of geometric objects. In this stage the programmer has full control since all executions are performed through the software. Which geometric objects can be displayed and what color can be assigned to the vertices of the corresponding objects are determined in the geometry stage. Therefore, mostly per-vertex or per-polygon operations take place in this stage. The geometric stage also consists of sub stages in the following order: *Model and viewing transformation, Lighting, Projection, Clipping* and *Screen mapping*. After having obtained the transformed vertices, colors and texture coordinates, the correct colors are assigned to the pixels in rasterization stage. For high performance graphics, it is difficult to change the implementation in the geometry and rasterization stages since they are mostly built on hardware.

Points, lines, and triangles are the rendering primitives that are used to build objects. Typically a scene consists of geometrically complex objects with different surface properties, such as texture, reflectance and transparency. The objects are rendered under various lighting conditions applying a local or global illumination model, which includes multiple, dynamically changing light sources and produces correct shadows. The rendering algorithm should be capable of resolving visibility of the objects and delivering images at interactive frame rates, even for very complex objects.

Before the 1980s, offline rendering was the main concern of computer graphics. Even though to create a photorealistic image could take long time to compute. However many applications require interactive 3D image synthesis. With the introduction of hardware support an important development has been achieved to improve the speed and quality of the interactive techniques. This opened a road to a new direction of research and development towards *real-time rendering*. Up to now, many algorithms have been developed to obtain higher quality and more realistic images. Over the past few years, the computational power of graphics processing units has changed dramatically. As a result a low-end PC graphics hardware now is capable to render millions of polygons shaded with very complex effects to create high quality images to render in real-time.

1.1.3 Overview

The research being covered in this report includes two separate topics namely *Hardware Based Rendering* and *Real-time Culling Techniques*. The core objective of this research is to review the developments of the methods within each of these topics and focus on the corresponding state-of-the art technologies.

In section 2, hardware based lighting and shading and in section 3, occlusion culling techniques are presented along with corresponding discussions and conclusions.

1.2 Hardware based lighting and shading

In this section we consider current graphics hardware. Graphics hardware can be accessed through a graphics application interface such as OpenGL [3] and DirectX [4].

In a graphics application one starts with a set of objects each of which comprises a set of graphical primitives. Also each primitive comprises a set of vertices. The collection of these primitives can be thought as defining the geometry of the scene. To process these vertices, a standard rendering pipeline with some variations is commonly used by graphics hardware [5]. In the beginning of the pipeline, a graphics processing unit (GPU) receives the geometry but usually points, lines and polygons are supported only. This information is passed one vertex at a time to the GPU, to perform vertex operations such as transformation and lighting. The projected geometry is scan converted, textured and piped through a number of tests such as depth and alpha tests. Finally resulting fragments are written to the frame buffer after blending these fragments with the existing ones. In such a pipeline every geometric primitive is executed in a fixed order. A major drawback with this type of pipelines is that it is difficult to modify the fixed order of operation and existing graphics hardware does not support such modification [6].

1.2.1 Vertex shaders

The two major functions of geometry stage are to carry out the transformation of the geometric primitives and to compute a color for each vertex. The graphic hardware designed according to fixed function pipeline supports one lighting model only which limits incorporating physically realistic models. After multiple stages of transformations the resulting geometry is transformed by a projection transformation and thus all primitives clipped and all vertices are transformed into screen coordinates. Texture coordinates are also defined in this stage.

Graphics architectures have experienced incredible changes in the last few years. It became apparent that the fixed function pipeline can not meet all the needs of 3D graphics programming. Recent developments in pipeline architecture has made possible to program the vertex processor. As a result of this using many important techniques in real time become possible [7].

The programmable part of the transform and lighting unit is called vertex shader which replaces the fixed function operations. A vertex shader is executed on each vertex as it moves along the pipeline. For each vertex, a vertex shader gets the vertex position defined by the application program and some related information including the current color, the texture coordinates and material properties. Every vertex shader outputs at least a vertex position for the rasterizer [1].

The vertex shaders support some basic instructions such as instructions for dot product computation, reciprocals and logarithms. The latest version of vertex shaders also can do simple branching [8].

1.2.2 Pixel shaders

The pixel shaders (or fragment shader) have similar functionality as the vertex shaders but the pixel shaders perform the execution after the rasterizer and operate on each fragment rather than on each vertex. Often pixel shaders require data from the vertex shader and they have to be "driven" by the vertex shader. For example to calculate per-pixel lighting the pixel shader needs the orientation of the triangle, the orientation of the light vector and in some cases the orientation of the view vector [9].

The pixel shaders execute a user defined program in which the instruction set works on four vector, operations include multiplication, dot product, reciprocals, square roots etc. Texture lookups are also possible with pixel shaders. Texture coordinates are computed in the fragment shader and then the lookup is done in the same shader. GeForceFX (NVIDIA) and Radeon 9700/9800 (ATI) cards are among the most popular high end products of pixel shaders.

1.3 BRDF-based lighting

In this section *Bi-directional Reflectance Distribution Function* (BRDF) is introduced briefly. A more detailed treatment of the subject can be found in a paper by Wynn [10].

A fundamental problem in computer graphics is how to model the light as it bounces off the object surface. An integral part of this problem is how to model the interaction of light with different types of materials. When light interacts with matter, a complicated light-matter dynamic occurs. This interaction not only depends on the physical characteristics of the light but also composition and the characteristics of the matter. For example, an opaque surface like cloth will reflect light differently than a smooth surface like a mirror differently. This reflection information can be modeled by a function called a *Bi-directional Reflectance Distribution Function* (BRDF). A BRDF is used during rendering to determine the appearance of a material under varying viewing and lighting conditions.

In graphics applications mostly simple models have been used to approximate reflection of light. However, these models (Phong model being the most sophisticated) can not meet the quality requirements of a wide range of graphics applications today. The use of techniques based on BRDFs, can drastically increase the visual realism of a scene [11].

When light strikes on a surface, some of it is reflected, some of it is absorbed and remaining portion of it is transmitted. It is well known that the light incident at a given surface is equal to the sum of reflected, absorbed and transmitted light. It is the reflected light that can be seen by an observer viewing the illuminated surface. A BRDF model describes how much light is reflected from a certain material.

The amount of reflected light depends on the viewing position and light direction relative to surface normal. Therefore a BRDF can be defined as a function of incoming and outgoing light directions. Another factor is the wave length of the incoming light. Finally, since the light interaction may not be homogeneous over the surface, positional variance is included in the BRDF model. A general BRDF can be written as $f_\lambda(\theta_i,\phi_i,\theta_o,\phi_o,u,v)$ where (θ_i,ϕ_i) and (θ_o,ϕ_o) represent the incoming and outgoing light directions in terms of spherical coordinates respectively, and (u,v) represent the surface position. Although positional variance is an important factor, it is common to use a BRDF without this factor. In this case the BRDF is called *position-invariant* and expressed as $f_\lambda(\theta_i,\phi_i,\theta_o,\phi_o)$. Positional variance can be introduced through the detail texture for a position-invariant BRDF. A BRDF is defined to be the ratio of the amount of reflected outgoing light to the amount of incoming light that is

$$f_\lambda(\theta_i,\phi_i,\theta_o,\phi_o)=\frac{L_o}{L_i\cos\theta_i dw_i} \tag{2.1}$$

There are two classes of BRDF: *Isotropic* and *anisotropic*. A material is called isotropic if its reflection is invariant with respect to rotation of the surface around its normal otherwise it is called anisotropic. The BRDF has to have two important properties: *reciprocity* and *conservation of energy*. Reciprocity property refers that the BRDF must be symmetrical in terms of the incoming and outgoing light. The conservation of energy property states that no more energy must be reflected than it is received [12].

The amount of outgoing light is a function of all incoming light and the BRDF at the surface point. This means that the amount of light reflected in the outgoing direction is the integral of such reflected light from each incoming direction. It is more convenient to use discrete space instead of continuous space since only limited number of light sources is used in practice. Based on this consideration, the general BRDF lighting equation for *n* light sources can be written as

$$L_o=\sum_{j=1}^{n} f_\lambda(\theta_i^j,\phi_i^j,\theta_o,\phi_o)L_i^j\cos\theta_i^j \tag{2.2}$$

Where L_i^j and (θ_i^j, ϕ_i^j) are the intensity and the direction of the *j*th light source respectively.

The Phong model without ambient light can be written as

$$L_o=L_i\left(k_d(\mathbf{L}\bullet\mathbf{N})+k_s(\mathbf{R}\bullet\mathbf{V})^n\right) \tag{2.3}$$

where **L** and **V** correspond to w_i and w_o in BRDF notation above, and **R** is the principle reflection direction. Rewriting this equation we get

$$L_o = \frac{Reflection\,(w_i, w_o)}{\cos\theta_i dw_i} L_i \cos\theta_i dw_i \qquad (2.4)$$

Comparing the equations (2.1) and (2.4), the following relation between BRDF and Phong models can be expressed as [10]

$$f_\lambda(\theta_i, \phi_i, \theta_o, \phi_o) = \frac{k_d(w_i \bullet \mathbf{N}) + k_s(\mathbf{R} \bullet w_o)^n}{\cos\theta_i dw_i} \qquad (2.5)$$

Unfortunately only a limited number of material reflectance properties can be modeled by this equation.

1.3.1 Implementing BRDFs using hardware

The easiest way of using a BRDF model is to compute the vertex color and pass it to the graphics pipeline. A major difficulty with this approach is that the BRDFs reflectance often does not exhibit a linear variation over the surface. This difficulty can be overcome by increasing the level of detail of the tessellated surface at the expense of increasing the computational cost. Vertex shaders can be used efficiently to compute the BRDF values. NVIDIA [14] has shown that for example, Minaert BRDF [15] can be implemented per vertex.

Pixel shaders are also used to compute BRDF models. Baodin [13] explains how to implement a Phong highlighting using a pixel shader. Computation of more complex BRDF models now is an active area of research [16]. Heidrich and Seidel [17] used two texture maps to approximate the corresponding terms in some illumination models. Kautz and Seidel [18] used pixel shading to render surfaces based on anisotropic representations.

Generally, BRDF models are composed of a series of functions and these functions are added or multiplied together. Complex illumination equations can be computed easily if the corresponding terms are treated separately. If these functions are expensive to evaluate on the shader they can be represented by textures.

Since a BRDF model except the simplified versions, has four parameters, the computed results can be represented in four dimensional (4D) tables. It would be nice to have a hardware accelerated 4D texture lookup table to store the BRDF and then compute the lighting equation performing a texture lookup for each pixel. The current graphics hardware has 2D texture mapping capability, no hardware currently available to support

4D texture mapping needed to implement a BRDF model. Storage requirements for the evaluated BRDFs would be very high even if 4D texture mapping were possible to implement.

One possibility of computing the lighting equation (2.2) is to make use of 2D textures to approximate the 4D function [19]. This can be achieved if the four dimensional BRDF can be expressed as the product of two 2D functions as

$$f_\lambda(\theta_i,\phi_i,\theta_o,\phi_o) = g(\theta_i,\phi_i)\cdot h(\theta_o,\phi_o)$$

Based on this factorization, a pair of 2D texture-lookups instead of a 4D lookups can be used to compute the BRDF function. Using this expression, the BRDF lighting equation (2.2) becomes

$$L_o = g(\theta_i,\phi_i)\cdot h(\theta_o,\phi_o) L_i co\theta_i$$

The function values corresponding to $g(\theta_i,\phi_i)$ and $h(\theta_o,\phi_o)$ can be obtained through 2D texture lookup.

The rendering process has two stages:

- Preprocessing stage: Factorizing 4D BRDF function into two 2D functions and representing as a pair of texture maps,
- Runtime stage: Reconstructing the BRDF end computing the lighting equation (2.2).

Wynn [19] provided an overview and implementation guide about how to separate and reconstruct the BRDF, and provide the corresponding algorithms.

For practical applications, it has been found that for many surfaces that just a single pair of two textures is sufficient to give reasonably convincing results for many materials [20]. The basic factorization algorithm and its implementation are given by Kautz *et al.* [21]. Examples of renderings of a teapot based on Kautz factorization algorithm is shown in Figure 1. McCool *et al.* [13] presented a new factorization technique with some advantages over the basic factorization algorithm.

1.4 Comparison of algorithms

In a recent paper, Kautz [7] have made a comprehensive review of local real-time shading algorithms. He has chosen 24 different algorithms and considered various BRDF models including diffuse, isotropic, anisotropic and shift-variant BRDFs. Clearly, the best choice of the algorithm depends on the effect to be simulated. He has concluded that algorithm that is developed by Kautz and Seidel [23] is the most general, flexible and most widely used.

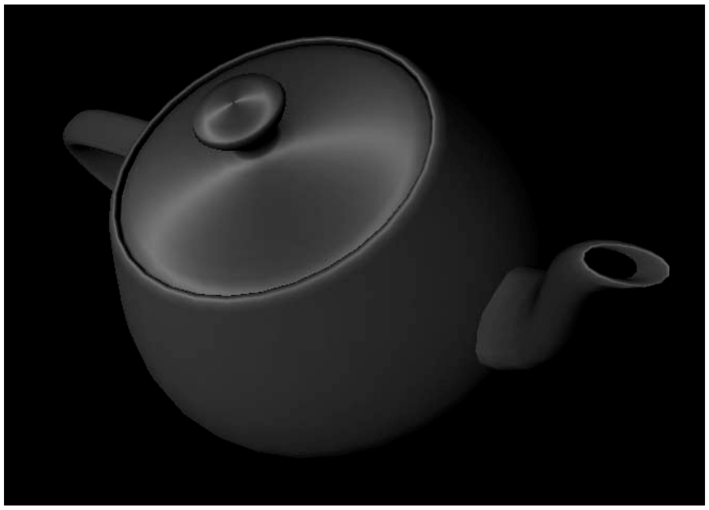


Figure 1. (a) Brushed Metal

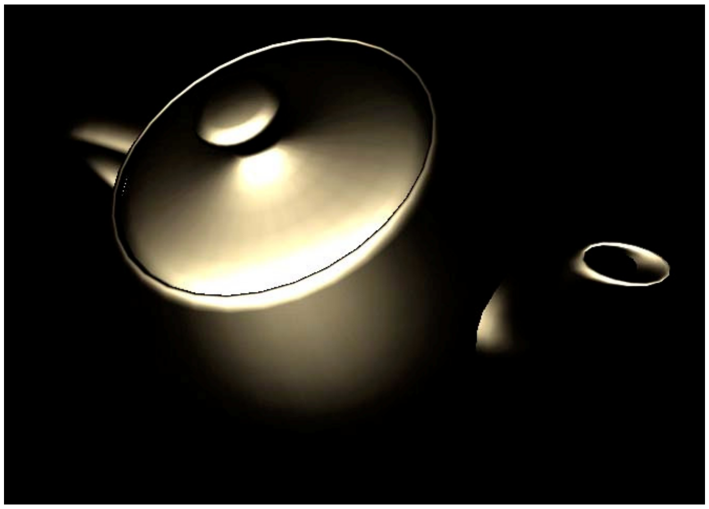


Figure 1(b): Gold

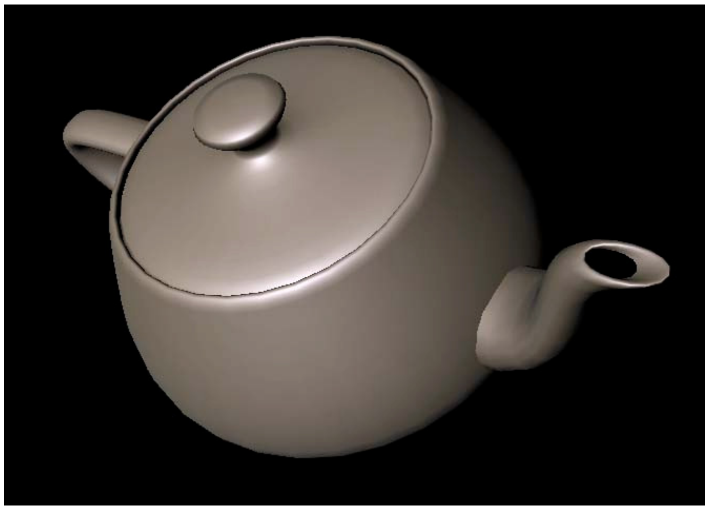


Figure 1(c): Pea-cock Feather

References

1. Angel E. Interactive Computer Graphics: A Top-Down Approach Using OpenGL. PEARSON-Addison Wesley, 2006.
2. Segal M., Akeley K. The OpenGL Graphics System: A Specification, 1999.
3. Neider J., Davis T., Woo M. *OpenGL: Programming Guide.* Addison Wesley, 1993.
4. Lindholm E., Kilgard, M., Moreton, H. A user programmable vertex engine. SIGGRAPH, pp 149-158, August 2001.
5. Kekoa P., William R. Mark, P H, Svetoslav T. A Real-Time Procedural Shading System for Programmable Graphics Hardware. Proceedings ACM SIGGRAPH 2001.
6. Anthony A. A., Larry G. Advanced RenderMan. Morgan Kaufmann, 2000.
7. Kautz J. Hardware lighting and shading: a survey. Computer Graphics 23 (1), pp 85-112, 2004.
8. Mitchell J. *Radeon 9700 shading.* ATI Technologies Inc., July 2002. An introduction to vertex and pixel shaders. http://www.dewmaster.net/articles/ shaders/
9. Wynn C. An introduction to BRDF-Based Lighting. NVIDIA Corporation, 2005.
10. Cole, F. H. Automatic BRDF Factorization. Thesis for the Bachelor of Arts, Harward College, Cambridge, Massachusetts, 2002.
11. Beckmann P., Spizzichino A. The scattering of Electromagnetic Waves from rough surfaces. McMillan, 1993.
12. Beaodin P., Guardado J. Non integer power function on the pixel shader. Shader X Wordware, May 2002. http://www.shaderx.com/ p226.
13. NVIDIA Corporation. NVIDIA SDK, May 2003. http://www.nvidia.com.
14. Minnaert M. Photometry of the Moon. In Planets and Satellites, pp 231-248. University of Chicago Press, 1961
15. Akenine T., Moller E.H. *Real-time Rendering.* A.K . Peters Ltd. Natick, Massachusetts, 2002.
16. Heidrich W., Seidel H.P. Realistic Hardware-accelerated Shading and Lighting. *Computer Graphics (SIGGRAPH 99 Proceedings),* pp 171-178, August1999.
17. Kautz J., Towards Interactive Bump Mapping. Anisotropic Shift-Variant BRDFs.
18. *ACM SIGGRAPH /Eurographics Workshop on Graphics Hardware,* pp 51-58, 2000. http://www.mpi-sb.mpg.de/~jnkautz/projects/anisobumpmaps/
19. Wynn C. Real-Time BRDF-based Lighting using Cube-Maps. NVIDIA White Paper, 2001 http://developer.nvidia.com
20. Kautz J., M.D. McCool M D., *Interactive Rendering with Arbitrary BRDFs using Separable Approximations*. 10th Eurographics Workshop on Rendering, pp. 281-292, June 1999

21. Kautz J., Wynn, C., Blow J., Blasband C., Ahmad E., McCool M. *Achieving Real-Time Realistic Reflectance, Part 2*.Game Developer, vol. 8, no. 2, pp. 38-44, February 2001.
22. McCool M D., Ang, J., Ahmad, *A. Homomorphic Factorization of BRDFs for High-Performance Rendering*. Computer Graphics (SIGGRAPH 2001 Conference Proceedings), pages 171-178, August 2001.
23. Kautz J., Seidel H.P., *Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs*. ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware, pp. 51-58, 2000.

2. Parametric Surface Rendering

Aydin Ozturk, Ahmet Bilgili, Serkan Ensoner
International Computer Institute, Ege University, May 2005
www.ube.ege.edu.tr
CADPIPE Internal Restricted Usage

Abstract

This section of the report presents methods which are used to design curve and surface shapes. Computer aided design (CAD) tools are widely used in industrial design of models. Today, Non-uniform Rational Splines (NURBs) become standard entities in CAD systems. A brief description of NURBs curves and surfaces is introduced. A number of important issues such as polygonal representation, tessellation and trimming the NURBs surfaces are considered and related work are reviewed. Algorithms developed for rendering large data sets are also discussed.

Keywords: *Parametric surfaces, NURBs rendering, NURBs tessellation, T-splines.*

2.1 Introduction

The world around us is full of different kind of objects and material surfaces such as clouds, trees, rocks, water, marble, plastic, metal etc. There is no common method to describe all these objects realistically. Nevertheless in computer graphics we tend to create the world with flat object. There is a good reason for such approximation approach. Graphics systems can render flat three-dimensional polygons at high rates, including doing hidden surface removal, shading and texture mapping.

Simple Euclidean objects such as sphere and ellipsoids can be described by polygon and quadric surfaces; natural objects such as mountains, clouds and fog can be modeled procedural methods like fractal techniques; in designing and modeling structures with curved surfaces such as aircrafts, buildings and car engines, spline techniques are commonly used.

Mainly, splines are used to design curve and surface shapes. A spline curve is specified by giving a set of coordinate positions called control points. Then a curve is fitted to these control points to interpolate or approximate the corresponding curve or surface.

There are a number of spline types including Cardinal Splines, Kochanec-Bartlet Splines, Bezier Splines and B-Splines that have been used to describe the curved surface. Among these, so called Nonuniform Rational B-Splines (NURBs) have been widely used. In the subsequent sections we explain various aspects of NURBs.

2.2 NURBs

A general expression for a point on a B-Spline curve is defined by

$$P(u) = \sum_{i=0}^{n} \mathbf{p}_i N_{i,d}(u) \tag{2.1}$$

where $\mathbf{p}_i$, (i=1,2, ..., n) represent the control points and the so called blending function $N_{i,d}(u)$ is the ith B-spline base function of degree d. These blending functions are defined by the Cox-deBoor recursion formulas [1]

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & u \in [u_i, u_{i+1}) \\ 0 & u \notin [u_i, u_{i+1}) \end{cases} \\ N_{i,d}(u) &= \frac{u - u_i}{u_{i+d} - u_i} N_{i,d-1}(u) + \frac{u_{i+d+1} - u}{u_{i+d+1} - u_{i+d}} N_{i+1,d-1}(u) \end{aligned} \tag{2.2}$$

Each blending function is defined over d sub-intervals defined by u_i ($0 \le u_0 \le u_1 \le \dots \le u_m \le 1$). u_i is called as a *knot* value and the entire set of selected values of u_i that is **u'=(**u_0, u_1, ... , u_m) is called as a *knot vector*. The flexibility of a B-Spline depends on the choice of the knot vector and the value of d which defines the degree of the corresponding polynom to be fitted.

If some weights w_i, (i=1,2, ..., n, $w_i > 0$) are used in (1.1) to assign more weights for certain control points, we obtain the following normalized representation

$$P(u) = \frac{1}{\sum\limits_{i=0}^{n} w_i N_{i,d}(u)} \sum_{i=0}^{n} \mathbf{p}_i w_i N_{i,d}(u) \tag{2.3}$$

This representation of B-Splines is called NURB. NURBS provide exact representation for quadratic curves such as circles and conics. They are also invariant with respect to perspective viewing transformation.

Formulation NURB surfaces is a straight generalization of the NURB curves that is

$$P(u,v) = \frac{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} N_{i,d}(u) N_{j,d}(v) w_{i,j} \mathbf{p}_{i,j}}{\sum\limits_{i=0}^{m} \sum\limits_{j=0}^{n} N_{i,d}(u) N_{j,d}(v) w_{i,j}} \tag{2.4}$$

By moving in the *u* and *v* directions and evaluating the equations for points on the surface a grid of the sample points on the surface can be obtained. A uniform triangle mesh can be obtained by joining the corner points of quadrilaterals.

The normal of each triangle surface can be computed by taking the cross product of surface derivatives

$$\mathbf{n} = \frac{\partial}{\partial u} P(u,v) \times \frac{\partial}{\partial u} P(u,v) \tag{2.5}$$

and normalizing the resulting vector. This evaluation scheme is known as *uniform tessellation* [2,3]. This method requires less CPU usage and easy to implement. Also it allows to precompute the blending function at the preprocessing stage [4]. Control points are invariant under affine transformations. Vertices are lighted in image space. A main property of NURBS is that the surface curve lies completely within the convex hull formed by its control points. Therefore the underlying polygon defined by the control points can be used as bounding box for culling. It has been shown that by repeatedly making subdivision through knot insertion the corresponding polygon based on control points converges to the surface [5].

A major difficulty faced in application of uniformly tessellated NURBS surfaces is that the underlying surface can be under sampled or over sampled. This problem has been considered in [6]. Adaptive tessellation approach approximates the surface more accurately when the surface has highly varying curvature but this technique is CPU intensive. Grahn *et al* [4] have reported that if uniform tessellation formed by dividing knot intervals into fixed number of subintervals can sample the surface accurately providing lesser computational requirements.

An important issue in forming the uniform tessellation of NURBs surfaces is determination of the step sizes. Some of the algorithms use size criterion determines the bounds based on the size of the resulting triangles in screen space. While others use deviation criterion to compute a bound on the maximum deviation of the tessellated surface from the NURBs surface algorithms in the second group generally produce good result but computationally expensive [3,7].

Joining the NURBs surfaces to model an object is not straightforward since the adjacent surfaces are not necessarily be tessellated with the same step size. Therefore cracks appear at the joining boundaries of the patches. If the boundary curve of each adjacent surface can be described by an identical parametric function then a strip of coving triangles can be obtained [3].

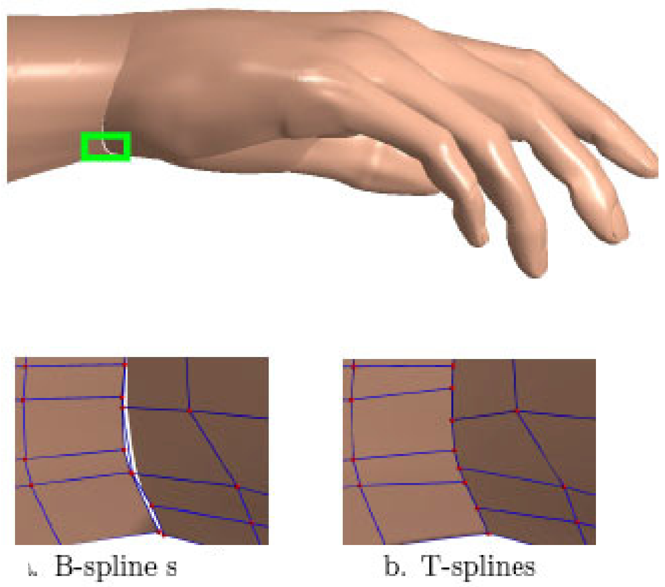


Figure 1. Hand model represented by spline surfaces. Cracks are seen in the small rectangular area: (a) Before sewing (b) After sewing.

In many applications, a surface modeling may require some features that can not be easily implemented. For example a hole on the surface is needed so that a conduit can pass through the surface. This can be achieved by so called trimming curves or trimming loops. Trimming curves are closed curves that can be used to render a surface from which the areas defined by curves have been removed. Trimming loops can be represented by

2D NURBS curves. The B-Spline representation can be used to render the tessellated trimmed surfaces [8]. The rendering algorithm computes the intersections of trimming curves with the isolines. In some algorithms [3,7] NURBs surfaces first are converted to Bezier surfaces since trimming operations are simpler in Bezier representations and then transformed back to NURBs representation.

Local control of NURBs curves and surfaces can be achieved by knot insertion. Such process does not alter the shape of the curve or surface. Various algorithms have been developed for knot insertion [9], [10], and [11].

In order to meet the high quality visualization demands posed by CAD applications, conventional NURBs rendering methods require a careful preparation of converted models. On the other hand the industrial need for larger and more detailed models is increasing; the CAD models are getting more complicated. For example a typical car body together with the visible parts of the interior consists about 70,000 trimmed NURBs patches with more than 5 million control points [12]. In such case the complete models often do not fit into the main memory and at once and therefore the use of out-of-core techniques become necessary. Guthe *et al* [12] developed a fast trimmed NURBS real-time rendering algorithm which has out-of-core support with fully automatic preprocessing.

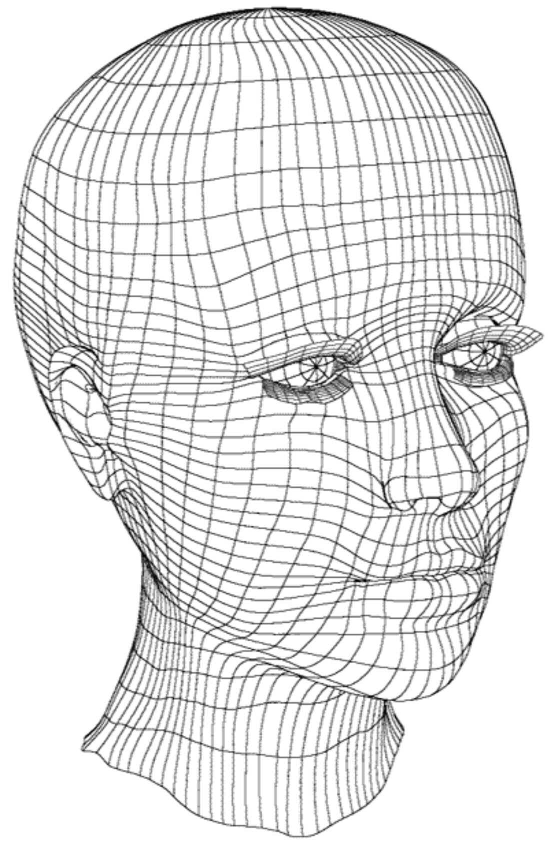


Figure 2: NURBS head model [15]

2.3 T-splines

For rendering purposes NURBs have been widely used in industrial design of models. However, there is a weakness with NURBs models that is it assumes that the corresponding control points lie topologically in a rectangular grid. As a result of this many control point have to be used only to meet the topological constrains but nothing else. Sederberg *et al* [13] introduced a class of splines which are generalization of NURBs, are capable to describe the same NURBs surface by removing the unnecessary control points. The difference between the number of control points used by the NURBs and the T-spline surfaces is depicted in Figure 3. In this example models based on T-splines used about one quarter of the total control points only. The T-spline model is geometrically equivalent to the NURBS model. Head models after applying NURBs and T-splines are shown in Figure 4.

It is important for designers to reduce the total number of control points by discarding the redundant control points. Redundant points also cause some undesired ripples in the spline surface.

2.4 Hardware Based Parametric Surface Rendering

The standard method for visualizing a parametric surface is to tessellate it (uniformly or adaptively) and render the triangulation. The rendering system performs the lighting calculation using vertex positions and normals, and linearly interpolates the color values across the triangles. Doggett *et al* [16] and Moule *et al* [17] both proposed a method for adaptive tessallation of parametric surfaces. Their method uses displacement mapping which is a technique for applying fine geometric detail to a simpler base surface, but requires a new hardware architecture which is not available yet. In a later study, Popa *et al* [18] proposed stream graphics architectures for adaptive tesellation.

A more advanced method for visualizing a parametric surface is to use pixel shaders, in order to linearly interpolate the surface normals (or other relevant quantities) instead of the colors, and do the lighting calculation per pixel. This strategy produces higher visual quality, but the correct color still can be obtained at the vertices only[19].

Another method is to attach to each vertex in the tessellation the associated pair of surface parameter values and utilize a pixel shader in order to (correctly) evaluate the surface positions and normals. This is possible for various types of surfaces, including B-spline and subdivision surfaces [20]. While this improves both the visual quality and the correctness, the problem is that only the vertices are rendered correctly. Therefore it is necessary to use a tessellation where each rendered triangle only covers a few pixels on the screen.

Yasui and Kanai [21] proposed a method for using the surface evaluation capabilities of modern graphics hardware for correct rendering of parametric surfaces. Their method renders a coarse tessellation of the surface using a pixel shader which evaluates the surface using the linearly interpolated parameter values, and "moves" the pixel into the correct position of the frame buffer..

Hjelmervik *et al* [19] proposed a new method inspired by Yasui and Kanai [21], which aims to remove artifacts produced by Yasui and Kanai's method. In addition, Hjelmervik's method ensures that the resulting image does not contain holes or other defects. The method proposed by Hjelmervik utilizes the graphics hardware to create a view-dependent tessellation, where each triangle covers approximately the same number of pixels[19].

2.5 Conclusion

NURBs as a generalization of B-splines have been widely used to describe the curved surfaces. They provide a convenient way to describe surfaces of almost any shape and the trimming process enables the designers to remove the unneeded areas corresponding to the joining hole on the surface.

NURBs surfaces have to be transformed into polygonal representation since current graphics hardware does not support direct rendering in their analytic representation with control points. In other words the NURBs surfaces need to be tessellated. After the triangulation of the NURBs surfaces are performed then the problem of rendering of these polygons is reduced to employing the most convenient lighting and shading models that are introduced in Section 1 of this report.

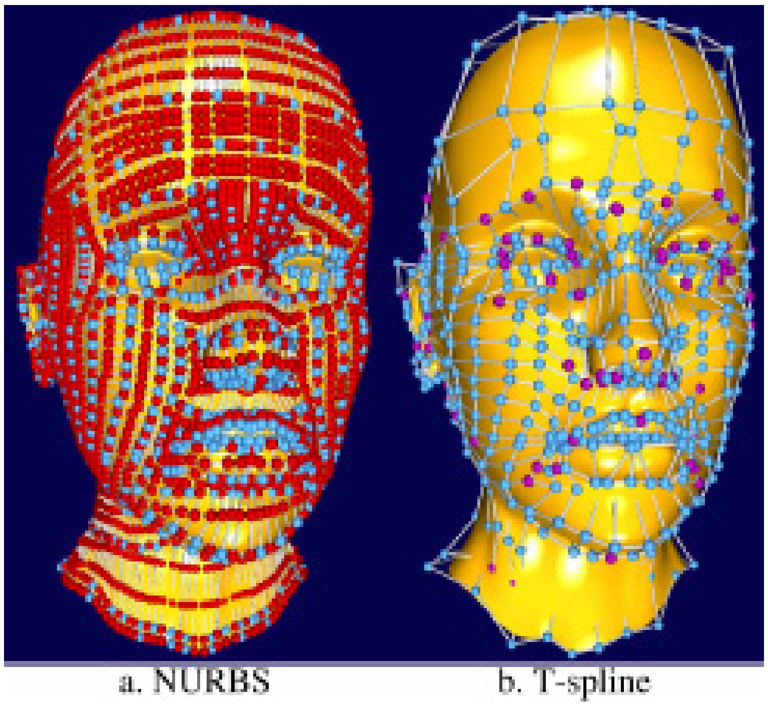


Figure 3. Head models based on NURBS and T-splines. (a) NURBs with 4712 control points (b) T-spline with 1109 control points (From Sederberg *et al* [14]). The red NURBs control points were not used in the T-spline model

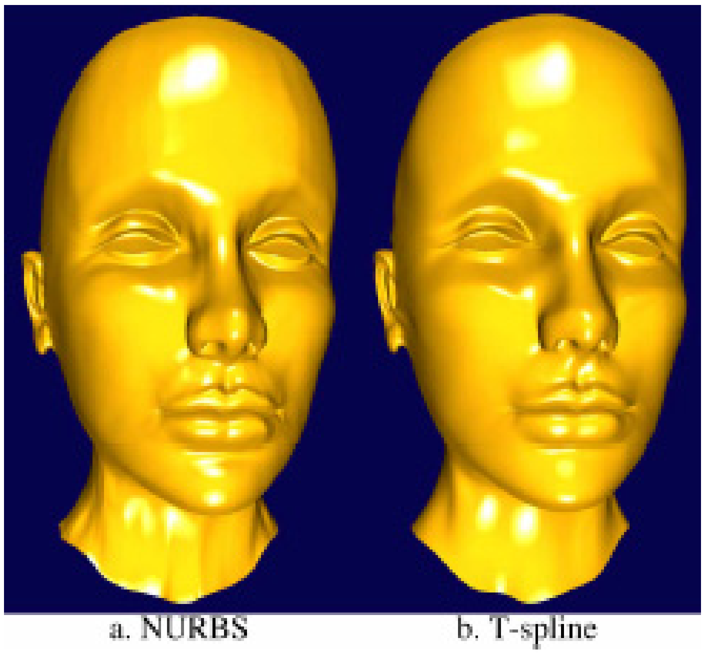


Figure 4. NURBs and T-spline models (From Sederberg *et al* [14]).

As the industrial need dictates the usage of more complicated CAD models based on large data sets, designers tend to use real-timeout-of-core techniques for rendering purpose. In this context, a *fast and memory efficient view- dependent trimmed* NURBs rendering algorithm developed by Guthe *et al* [12] can be used.

T-spline techniques as a new way of describing surfaces seem to be promising and it has a potential to be the state-of-the art technology in the near future. Currently it is hard to make a detailed evaluation of this technology as it does not have an open source.

References

1. Hearn D., Baker P.M. Computer Graphics with OpenGL. *Pearson Prentice Hall,* Pearson Education, Inc., NJ, 2004
2. S. Abi-Ezzi and S. Subramaniam, Fast Dynamic Tessellation of Trimmed NURB Surfaces, Computer Graphics Forum, 13(3), 107---126, 1994. (Eurographics '94.)
3. Kumar S. Manchu D. Astral A. Interactive display of large scale NURBs models. IEEE Transactions on visualization and Computer Graphics, 2 December 1996.
4. Grahn, H. Volk T., Wolters H.J. Nurbs in virtual VRML.ACM 2001 .
5. Dahmen W. Subdivision algorithms converge quadratically. J. Comp. Appl. Math., 16, pp125-158, 1986.
6. Peterson J.W. Tessellation of NURBS surfaces. Graphics Gems IV, Academic Press pp 286-320, Boston 1994.
7. Rockwood A. Heaton K. Davis T. Real-time Rendering of trimmed surfaces. In Proceedings of ACM Digraph, pp107-117, 1989
8. Lukens W.L., Cheng F. Tessellation trimmed NURB surfaces. Computer Science Research Report 19322(84059), IBM Research Division, 1993.
9. Cohen E., Lyche, T. Riesenfeld, R.F. Discrete B-spline subdivision techniques in computer aided and computer graphics. Computer Graphics and Image Processing 14 pp 87-11, 1980
10. Bohem W. Generating the Bezier points of B-spline curves and surfaces. Computer aided Design 13, pp 365-366, 1981.
11. Seidel H.P. Knot insertion from a blossoming point of view. Computer Aided Geometric Design 5, pp 81-86. 1988.
12. Guthe M., Balazs A., Klein R. Real-time-out-of core trimmed NURBS rendering and editing, In proceedings of Vision, Modeling and Visualization 2004, Akademische Verlagsgesellschaft Aka GmbH, Berlin, ISBN 3-89838-058-0, November 2004, pages 323-330.
13. Sederberg T.W., Zheng J., Bakenov A., Nasri, A. T-Splines and T-NURCS. ACM Transactions on Graphics 22, 3, 477-484 (2003).
14. Sederberg T.W., Cardon D.L., Zheng J. Lyche T. T-Spline simplification and local refinement, ACM Transactions on Graphics, 23(3), 2004.
15. http://www.3drender.com/jbirn/3drender.com/jbirn/ea/HeadModel.html
16. Doggett, M., and Hirche, J. Adaptive View Dependent Tessellation of Displacement Maps, in: Proc. of Eurographics/SIGGRAPH workshop on graphics hardware 2000, pp. 59-66, 2000
17. Moule, K., McCool, M.D. Efficient Bounded Adaptive Tessellation of Displacement Maps. Graphics Interface 2002: 171-180
18. Moule, K., Popa, T.S. and McCool, M.D. Stream GPU Architectures. Technical Report CS-2003-23, School of Computer Science, University of Waterloo, August 2003

19. J. M. Hjelmervik and T. R. Hagen. GPU-based screen space tessellation, in Mathematical Methods for Curves and Surfaces: Tromsø, 2004, M. Dæhlen, K. Mørken, and L. L.Schumaker (eds.), Nashboro Press, 2005
20. Bolz. J., and P. Schroder, Evaluation of subdivision surfaces on programmable graphics hardware, submitted for publication, 2003.
21. Yasui, Y., and T. Kanai, Surface quality assessment of subdivision surfaces on programmable graphics hardware, in Proc. International Conference on Shape Modeling and Applications 2004, IEEE CS Press, Los Alamitos, CA, 2004, 129–136.

3. Reduction: State-of-the-Art

Sophie Jarlier, HyungSeok Kim, Stephane Garchery, Nadia Magnenat-Thalmann
MIRALab, University of Geneva, May 2005
www.miralab.ch
CADPIPE Internal Restricted Usage

Abstract

The complexity of 3D models grows faster than the ability of our hardware to render them. Since the first structure described by Clark [Clark1976] incorporating LOD and other common techniques such as view-frustum culling, many ameliorations have been exploited. We describe here the types of models to simplify and the reasons of simplification. This process of mesh simplification is based on a geometric or topological approach, using off-line or in runtime an error measure that controls the quality of the results. Then we overview the different methods in this field according to the approach they use and we explain the need of appearance preserving and out-of-core techniques.

Keywords: *data reduction, polygonal simplification, multiresolution modeling, surface approximation, level of detail, decimation, refinement, view-dependent rendering*

3.1 Preliminaries

3.1.1 Simplification for what?

3.1.1.1 Real Time rendering (visualization in RT)

In many graphics domains the speed is the main importance. For example in video games it is important to perceive in real-time the scene changing while you are navigate inside the game world. A CAD designer wants to visualize rapidly the results of his modifications on the model without having to wait several hours of computation. For these reasons, it is important to find techniques to simplify the models to be able to see them in real-time. The creation of various levels of detail (LOD) associated to a model will allow to visualize only the part necessary to the user. For example, when the object is far away, less detail will be needed as human eye can not perceive these details and in the opposite more details should be display when the viewer is closed from the object (see

Figure 1). Less detail is needed to be computed for visualization, less time we will require to display it.

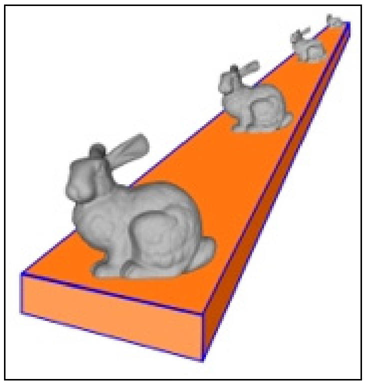


Figure 1: An example of levels-of-detail [Luebke2001]

3.1.1.2 Transmission to the network

Many researches have been focused on the extremely high detail models of the scanned 3D shapes. The scanning technology greatly improved both the quality of model and the size of data to be processed. Thus it is necessary to find solution to transmit and render huge datasets to the network. Many approaches are presented to reduce the complexity of details (usually in polygons) to transmit without loosing the quality of the model.

3.1.2 Simplification to what?

Simplification algorithms are generally applied to mesh objects (most of the time triangulated) or curve objects.

3.1.2.1 To curves

Curves open and closed, are 1-dimensional manifolds. The first idea to represent a curve is to store a curve as many small straight line segments. However, as the smoothness of the curve increases with the number of points/line segments, it is not convenient to have to specify so many points. Another approach to represent the curve is to work out the equation that represents the curve but it is difficult for complex curve and moving an individual point requires re-calculation of the entire curve. To resolve these problems, is used the interpolation. Then the curve is obtained by defining a small number of points and use interpolation to invent the extra points for us and joining the points with a series of straight lines. The representation of straight line can be done using linear parametric equation, while curves need polynomial equations. Polynomial curves can be represented by the sum of polynomials (called also basis functions) each weighted by a 2D or 3D coefficient control (the control points) as follow:

$$\mathbf{Q}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \sum_{i=0}^{n} B_i^n(t) \cdot Pi$$

Where $\mathbf{Q}(t) \in \Re^3$ is a point on the curve,

$Pi \in \Re^3$ are the control points,

and $B_i^n(t) \in \Re$ are the basis functions.

The shape of the curve depends on the degree n:

- If n=0, the curve is constant
- If n=1, the curve is linear
- If n=2, the curve is quadratic
- If n=3, the curve is cubic

In this document, focus is made on the representation of 3D shapes. Thus, further detail on the curve representation and corresponding parametric surfaces is covered in other deliverables.

3.1.2.2 To meshes/surfaces

A mesh is a collection of triangular or quadrilateral polygons (i.e. closed plane figures formed by three or more line segments) that forms a surface. This graphics object is composed of vertices, edges and faces that represent a topological polyhedron (i.e. a solid figure with flat faces that are polygons).

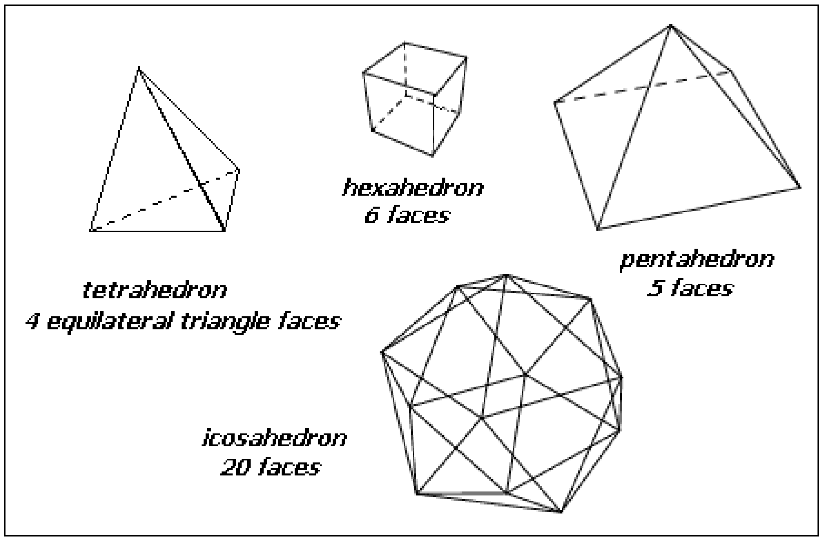


Figure 4: Examples of polyhedrons

The **topology** is the properties that are preserved through deformations, twists, and stretches of objects. Thus the topology of the mesh is preserved if simplification does not result in creation or deletion of holes. While a cup and a sphere are topologically equivalent, a sphere and a doughnut are not.

A mesh has two components:

- The mesh geometry that is represented by vertices

- The mesh connectivity that is represented by edges or faces that connect vertices.

The mesh connectivity encodes the topology of the mesh. Simplification of the mesh geometry may or may not result in simplification of the mesh topology.

Most of simplification algorithms deal exclusively with triangle meshes. If the mesh is not composed of triangulated polygons, it would be necessary to triangulate them at a preprocessing step.

The vertices of a mesh can be classified [Schroeder92] into five categories:

- **Simple:** a simple vertex is locally manifold surrounded by a single complete ring of triangles, each of which shares a single edge with the vertex
- **Boundary:** boundary vertices differ in that the set of triangles does not form a complete ring
- **Interior edge:** a simple vertex with less than three feature edges (defined by angle between two triangles but which could also depend on material attributes such as color or texture coordinates)
- **Corner:** vertices with three or more feature edges
- **Nonmanifold: vertices which make nonmanifold surface parts.**

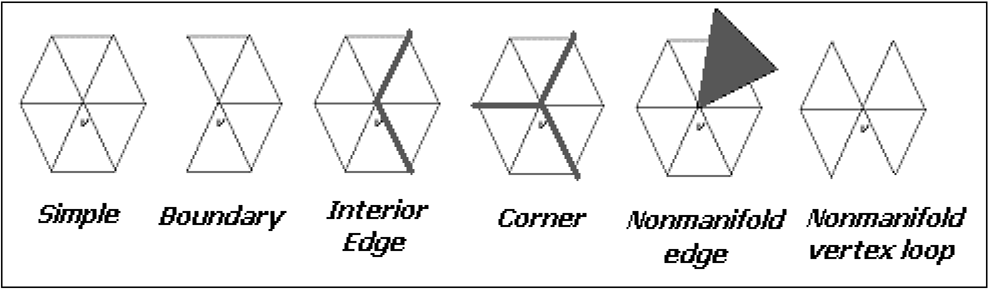


Figure 5: Types of vertices

The simplification method have been considering some or all of those different types of vertices as it could be a important features to the shape or constraints given by user. Dealing with those constraints could be dealt in explicitly or implicitly.
Different kinds of meshes have been chosen to be simplified such as height fields, manifold or non-manifold surfaces that we describe in the next sections.

3.1.2.3 Height/scalar fields and parametric surfaces

Height fields and parametric surfaces are the simplest class of surfaces. They can be defined as a set of points satisfying an equation of the form z = f(x,y) where x and y range over a subset of the Cartesian plane. They can also be defined as a matrix of points that are distributed regularly on a two-dimensional grid. While height fields are generally used to create terrain models of hundreds of triangles in a mesh, parametric surfaces are used to construct NURBS for example.

3.1.2.3.1 Manifolds

Manifolds are more difficult to simplify than parametric surfaces or height fields because there is no natural 2D parametrization of the surface.

Manifolds (and only manifolds) can be defined by:

- Their **genus** g which represents the number of holes in the mesh.
  Examples: a sphere and a cube have g=0 while a torus or a coffee cup have g=1
- Their **Euler characteristic** e = V-E+F with V: vertices, E: edges and F: faces. So the relationship between e and g is: e = 2-2g
  Examples: sphere or cube have e=2-2(0)=2 while torus has e=2-2(1)=0
- Their **orientability**: if we have consistent normal direction for a point then the object is orientable otherwise non-orientable.

3.1.2.3.1.1 2D Manifolds

2D manifolds follow these properties:

- every edge has exactly two incident triangles
- neighborhood of every point belonging to the object is homeomorphic to an open disc
- thus, every edge is bordered by exactly two faces; t-junctions, internal polygons, and breaks in the mesh are not allowed

While 2D manifolds with boundaries follow these properties:

- every edge has either one or two incident triangles
- neighborhood of every point belonging to the object is homeomorphic to an open disc or a half-disk

3.1.2.3.1.2 3D Manifolds

3D manifolds follow these properties:

- every triangle has exactly two incident tetrahedrons
- neighborhood of every point belonging to the object is homeomorphic to an open ball

While 3D manifolds with boundaries follow these properties:

- Every triangle has either one or two incident tetrahedrons
- Neighborhood of every point belonging to the object is homeomorphic to an open ball or a half-ball

3.1.2.3.2 Non-manifolds

Non-manifold surfaces are the most general class of surfaces which permits three or more triangles to share an edge and permits arbitrary polygon intersection. In non-manifold surfaces can appear edges shared by more than two triangles or patches touching each other in one vertex. (see Figure)

3.1.3 Survey of polygonal reductions

3.1.3.1 Culling and caching

The culling method reduces costs for the entire rendering pipeline. The culling method eliminates a portion of the data that is not seen in the final rendered images. By eliminating the data before the rendering pipeline processing, the whole pipeline stage benefits. This method is especially useful for systems with surplus processing power in the application processor, which is the case of most current rendering systems. The culling method is described in other STA documents, thus it will not be covered in this document.

The caching method is also a very effective and popular method to improve rendering performance. Basically, caching methods try to cache information to relieve the burdens at bottleneck stages. By reducing the time cost of a bottleneck stage, the entire performance of the pipeline is increased. In modern rendering systems, high-speed rendering subsystem cache memory is used to cache information to be transferred through the bus or to cache intermediate computation results of the geometry processor. For geometry, most current rendering API provides control over the caching mechanism. Geometry data can be cached in the cache memory, and intermediate computation results are stored in a special form of ‘draw lists’.

3.1.3.2 Multiresolution/Mesh simplification

While height fields (called terrains in cartography) and parametric surfaces use simplification methods such as regular grid, hierarchical subdivision, refinement and decimation methods, mesh simplification use specific operators and error metrics. These multiresolution techniques will be described in section III.

3.1.3.3 Geometric compression

It is also important not to confuse the geometric simplification and the geometric compression. Indeed these two techniques are greatly different and we deal in this paper only with simplification methods. Just for a brief description of geometric compression, we can define it as a coding of meshes where the vertices of the mesh are coded in an order that partially contains the topology of the mesh. **Taubin and Rossignac** [Taubin1998] were the first to propose to use the transmission order of the mesh vertices to code the topology and code the vertex positions. **Touma and Gotsman** [Touma1998] described a similar technique but with a different way to traverse the triangulation. And **Cohen-Or, Levin and Remez** [Cohen-Or1999] used multiresolution decomposition to obtain a progressive compression of the meshes.

3.1.4 Conclusion

Since the progresses of science in term of design modeling and thus the increase of data sets size, mesh simplification became a real need to transmit these models over the network while keeping fidelity and having the possibility to visualize them in real-time.

These kinds of models can be from different types and the choice of the algorithm will differ with the models treated.
In this document, main focus will be given to the triangular mesh as it is the main representation scheme for the 3D visualization. Another important representation is the parametric surface which is quite popularly used in CAD domain. The useful feature of the parametric surface such as NURBS is that the simplification process is usually conducted through controlling tessellation levels. Furthermore, there have been a few approaches to devise and use hierarchical parameterizations such as hierarchical B-Spline, Wavelet, etc. So far, it is quite time consuming task to render those parameterized surfaces in real-time, the most of approaches finally converts the parameterized surface into triangular meshes.
This document is composed of 4 main parts, in Chapter 2, evaluation mechanisms to measure the fidelity of simplified model are described. The actual simplification and multiresolution methods are described in the Chapter 3 and the applications and remaining issues are presented in the Chapter 4 and 5.

3.2 Evaluation

3.2.1 Simplification error evaluation: measurement of the output quality

3.2.1.1 Definition: geometric opposite to perceptual evaluation

The simplification process consists in a process of obtaining a simplified triangulated mesh M’ as similar as the original mesh M. For many algorithms, this similarity is denoted as an error. In order to evaluate the accuracy of the approximated mesh obtained and consequently evaluate the differences between the two meshes, researchers need to have a precise evaluation of the error of this simplification or a tight and guaranteed upper bound on the error that will guide the selection of the appropriate level of detail that meets the desired graphics fidelity during the simplification and rendering processes. The human visual system and human intuition is not sufficient for this kind of error measure because not appropriate for any LOD systems. The problem to which the researchers are confronted is the definition of this similarity. This similarity can be defined in terms of geometry (shape of the two meshes, viewing conditions such as size and orientation of the shape on the screen) but also in terms of appearance (color, texture, normals).

The error metrics can be used along the simplification and rendering process or after simplification to evaluate the quality of the results. In this last case, a tool is available Metro which is overviewed later in this paper. Nevertheless it can be more appropriate to measure the error during the simplification process when we generate a large number of levels of detail as in the case of a hierarchy for view-dependent adaptation.
Two kinds of approaches can be employed during the simplification process according to the researched goals:

- Minimizing the number of polygons: given an error bound ε, the goal is to minimize the size of output mesh, which is called fidelity-based simplification by Luebke et al in his survey [Luebke2002]

- Minimizing the approximation error: given an expected size of the mesh, the goal is to minimize the error ε, which is called budget-based simplification by Luebke et al in his survey [Luebke2002]

3.2.1.2 Approaches for estimating simplification errors:

3.2.1.2.1 Locally bounded error

Local bounded error approximation is used when the approximation accuracy is known around each surface entity. Most of decimation methods are using this error approximation.

3.2.1.2.2 Globally bounded error

Global bounded error approximation is used when the approximation accuracy is known only for the entire simplified mesh. This approximation error is employed in several simplification techniques such as simplification envelopes, superfaces, clustering approaches, and methods based on the conversion into an intermediate hierarchical representation.

3.2.1.2.3 Image-space error evaluation

Previously mentioned error metrics are mainly concerned with similarities in geometric space. The visual degradation can be controlled taking care of preserving curvature and sharp edges which gives a good control of the appearance of the shape. Pictorial information (color and texture) are from a primordial important factor in perception. Color discontinuities are managed carefully by Reddy, Hoppe, Certain and Soucy.

3.2.1.3 Tools for estimating simplification errors:

3.2.1.3.1 Geometric approximation error

The goal of geometric error is to preserve the shape of the original mesh as best as we can. The Euclidian distance allows the measure of distance between two points p1=(x1,y1,z1) and p2=(x2,y2,z2) and is defined as follow:

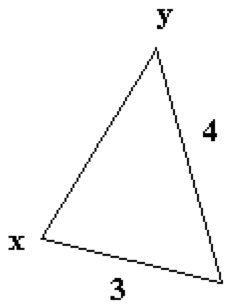
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Three norms exist: L1, L2 and L∞. Each one can be defined by:

$$\|x_p\| = \sqrt[1/p]{\sum_i |x_i|^p}$$

With p=1 for the L1 norm, p=2 for the L2 norm and p=± ∞ for the L∞ norm.
Let’s take an example: if we have the following triangle, the distance between x and y is measured as follow:

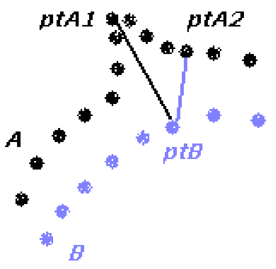
- in L1 norm: dist(x,y) = 4 + 3
- in L2 norm: dist(x,y) = $\sqrt{4^2+3^2}$
- in L3 norm: dist(x,y) = $\sqrt[3]{4^3+3^3}$ cubic
- ...
- in L$_\infty$ norm: dist(x,y) = Max(4,3) or Min(4,3) according if it tends to –∞ or +∞



3.2.1.3.1.1 L$_\infty$ norm: Infinity norm or Chebyshev norm or Hausdorff distance

This norm calculates the maximum of pointwise distances. It can be useful if focusing on the maximum error occurring in the solution. It is defined by:

$$E_{\max} = \max_{i=1}^{k} |x_i - y_i|$$



The closest point from A includes the closest point from ptA1 which is ptB. Also the closest point from B is inferior: the closest point from ptB is ptA2.

Given two points set A and B, the Hausdorff distance measure consists in finding the closest point in set B for each point of A and vice versa. The Hausdorff distance between a mesh S and its approximated mesh S' is defined as follow:

$$H(S,S') = \max(dev(S,S'), dev(S',S))$$

where:

$$dev(A,B) = \max_{a\in A}(dist(a,B))$$

$$dev(a,B) = \min_{b\in B}(\|a-b\|)$$

$$dev(A,B) = \max_{a\in A}\min_{b\in B}(\|a-b\|)$$

This distance measures the worst case distance that a point on one surface would have to travel to reach the other surface. This distance is commutative: H(A,B) = H(B,A). But dev(A,B) ≠ dev(B,A). Hausdorff provides a bound on the maximum geometric deviation between the two shapes and hence provides the maximum distance between a point on a profile (= visible silhouette, the visible edges) and the profile of another. The

profile depends on the view but not the Hausdorff distance. If Hausdorff distance between S and S' is 0 then S and S' have an identical shape.
Hausdorff is the tightest possible bound on the maximum distance between two surfaces. However, Hausdorff measure is not a good measure neither of shape similarity neither on proximity of surface orientation between two shapes. The cost of computing this distance between two polyhedra is significant because it is not sufficient to check the distance between all vertices of one set and the other set, one must be able to detect the configuration where the Hausdorff distance is realized at points that lie in the middle of faces.

3.2.1.3.1.2 L1 norm: Manhattan distance or city block norm

This norm calculates the average of the error (average of the pointwise distances) in the solution domain in case of cubical cell for example. As p=1, this distance can be defined by:

$$E_{avg} = \sum_{i=1}^{k} |x_i - y_i|$$

3.2.1.3.1.3 L_2 norm: Euclidean distance and root mean square

The L2 norm or Euclidean distance is used to measure a distance (length of a vector), an energy (energy in an image) or an error (amount of error or distortion between an original signal or image and its reconstruction) in case of spherical cell for example. It is defined by:

$$d = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

Closely related to this L2 norm, the root mean square error (RMS) is normalized by the number of elements in the vector or signal. It is based on the principle that each pixel of a monochrome image is represented by a single number indicating luminance. When I is the original image and I' is the simplified image, the root mean square error is defined as:

$$E_{RMS} = \sqrt{\frac{1}{N} \sum (I_{xy} - I'_{xy})^2}$$

Where: I_{xy} and I'_{xy} are pixels from the original and simplified images. This digital measure is fast and simple; however it leads to some imperfections associated to its response to visual differences such as visual artifact. It is important to note here that the

RMS distance can be normalization by N of the Euclidean distance but of the Hausdorff distance as well.

While the maximum error provides a guaranteed error bound (knowing that the error is never more than some specified tolerance), the average error can be an indication of the error across the entire surface as opposed to a few particularly bad locations (maximum error can be 10 times larger than the average error for many models). The ideal is to show a compromise between both, bounding the maximum error without allowing the average error to grow in uncontrolled fashion -> really rarely used

3.2.1.3.2 Non-geometric error / Attribute errors / Appearance approximation error

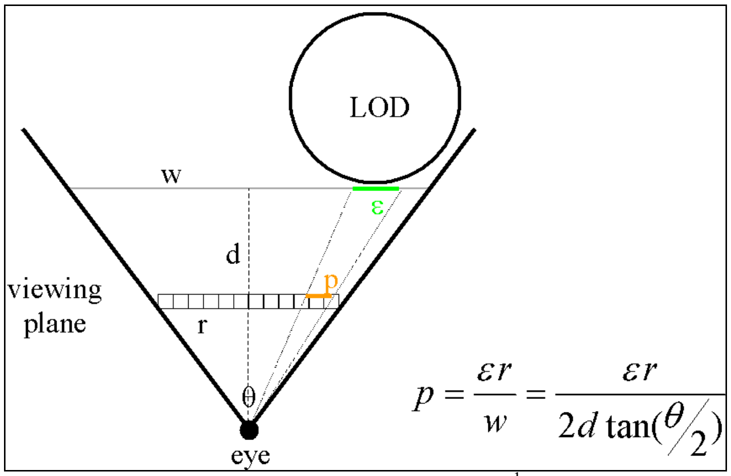
3.2.1.3.2.1 Appearance: Screen-space error

Last error measure tools are used to measure the surface deviation in a 3D object space. However, some parameters such as viewpoint, field of view, etc. can be taken into account to measure error during the rendering. In the figure 6 below:

- **θ** is the total field of view
- **d** is the distance in the viewing direction from the eye point to the level of detail LOD (or its bounding volume)
- **r** is the resolution of the screen in pixels
- **w** is the width of the viewing frustum at distance d
- **p** (in orange) represents the screen space deviation measured in pixel units
- **ε** is the length of an error vector.

This technique gives an approximation to the projected size of the error in pixels. The LOD with the smallest number of triangles to render for the error tolerance t≥p is rendered, i.e. the LOD whose ε is as large as possible with:

$$\frac{\varepsilon}{w} = \frac{p}{r} \quad \text{and consequently} \quad \varepsilon = \frac{pw}{r} = \frac{p \cdot 2d \tan(\theta/2)}{r}$$


LOD
w
ε
d
p
viewing plane
r
θ
eye
$p = \frac{\varepsilon r}{w} = \frac{\varepsilon r}{2d\tan(\theta/2)}$
Figure 6: Screen-space error[1]

3.2.1.3.2.2 Color

The colors of a polygonal model are stored as (r,g,b) triples with each value in the range [¨0,1]. We then consider the RGB space: red, green, and blue form the orthogonal basis vectors of the coordinate system. We measure the color error in the RGB space as in a Euclidian space by computing the RGB distance between corresponding points as:

$$d_{color} = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

The optimized color value of a simplified vertex may lie outside the valid color range. Clamping the value to [0,1] keeps it in range but increases the color error.

3.2.1.3.2.3 Texture coordinates

Texture coordinates for polygonal surfaces are represented as (u,v) coordinate pairs that define a mapping of vertices to points in a 2D texture space. This texture space uses values in the range [0,1]. Thus the same issues with optimized values going out of range apply, and clamping remains a viable solution. The difference with color here is that the texture coordinates are intended to describe a bijection (one-to-one and onto mapping) between the polygonal surface and the texture space.

3.2.1.3.2.4 Normal

The natural space for normal vectors is on the Gaussian sphere, a unit radius sphere centered at the origin, on which each point represents a normal vector. The measure for distance between two normal vectors is an angular distance:

[1] J. Cohen. Measuring Simplification Error. Slides talk

$$d = ar\cos\left[\begin{pmatrix} n_{1_x} \\ n_{1_y} \\ n_{1_z} \end{pmatrix} \cdot \begin{pmatrix} n_{2_x} \\ n_{2_y} \\ n_{2_z} \end{pmatrix}\right]$$

3.2.1.3.3 Combining approach

A simplification algorithm can measure the geometry error plus other attribute errors such as color, texture and normal during the evaluation process of the simplified mesh. For a given operation, we need to weight the error terms for the purpose of prioritization of simplification operations.

3.2.1.3.4 The Metro tool, an implemented tool to measure surface distance

A tool for the evaluation of the error introduced during the simplification process was needed. For this reason, Cignoni et al [Cignoni1998] created the Metro tool [MetroTool] which allows the comparison between surfaces. Based on point-surface distance measure (see Section III/B/2/c), this tool evaluates the output quality of simplification algorithms.

- Input: Possible inputs for this tool are OpenInventor formats and raw indexed representation (list of vertex coordinates, list of triangular faces defined by three indices to the vertex list). Two polygonal models and a sampling distance. One of the input models is sampled by points spaced at the specified distance.
- Outputs: Based on surface sampling and point-to-surface distances, Metro returns both numerical and visual evaluations of the meshes' likeness
    - Numerical: Numerical outputs contains information on the topology, the size, the surface area, the mesh volume, the feature edges total length, the diagonal of the minimal bounding box, the diameter of the minimal bounding sphere. Metro computes the maximum and average distance from the first model to the second.
    - Visual: The error is visualized by coloring the pivot mesh with respect to the evaluated approximation error in order to show how distance function is distributed across the model. Two different kinds of colors mapping are available: the *per-vertex* mapping that assign a color proportionally to the error on each mesh vertex and the *error-texture* mapping that computes a RGB-texture for each face and stores the color-coded errors evaluated on each sampling point. To these visual representations, are associated a histogram.

3.2.2 Empirical evaluation of simplification code

Cignoni et al proposed an evaluation of some simplification codes using their Metro tool [Cignoni98].

The models used for these tests were:

- Bunny [Bunny]:

    - Model of a plastic rabbit scanned
    - 34,834 vertices
    - 69,451 triangles
- Fandisk [Fandisk]:
    - Valid representation of a CAD models
    - CAD models have per-vertex normals and a single color
    - Sharp edges and sophisticated surface curvature
    - 6,475 vertices
    - 12,946 triangles
- Femur [Femur]:
    - Isosurface from a CT scan of a human femur
    - 76,794 vertices
    - 153,322 triangles

And the codes evaluated were the following available public domain codes which will be described later in this paper:

- Mesh decimation (Schroder et al 1992)
- Simplification envelopes (Cohen et al 1996)
- Multiresolution decimation (Ciampalini et al 1995)
- Mesh optimization (Hoppe 1993)
- Progressive meshes (Hoppe 1996)
- Quadratic error metrics simplification (Garland and Heckbert 1997)

Their conclusions can be found in their survey written in 1998 [Cignoni98]. Simplification envelopes are the method which takes the most time, while quadric error metrics and mesh decimation are the methods that take the least. The best technique at preserving area is the progressive meshes while the worst are the simplification envelopes or mesh decimation. Multi-resolution approach is the one that uses the least memory while simplification envelopes use a large memory.

3.2.3 Performances: Speed complexity and fidelity

The performances of the simplification algorithms can be expressed in term of speed (time necessary for the computation and the real-time rendering), complexity (the resulting reduced number of polygons in the simplified mesh in comparison to the initial number of polygons), and fidelity (the quality of the simplified mesh obtained in terms of visual artifacts etc.).

As for each scientific research, evaluation is needed to prove the quality of the results. The simplification algorithms must be evaluated in different terms: the fidelity of the model, the speed of the algorithm, the storage space needed and the simplification code optimization. The key element of a simplification algorithm is surely the error metric which compares the simplified mesh to the initial one using some error metrics controlling the geometry and topology of the mesh as well as the appearance preservation of the mesh. This evaluation can be done once the simplification process has terminated or at runtime. The Metro Tool aimed at giving researchers a common meaning of evaluation of their work, however it can not be used at run-time.

3.3 Mesh Simplification Approaches Overview

3.3.1 Introduction

3.3.1.1 Classification of methods

The problem of the elaboration of this state-of-the-art is how to classify the multiresolution approaches. Basically these methods can be sorted in three categories: the simplification methods using operators based on geometric approaches, the simplification methods based on error metric and the ones using hierarchical structures. But most of these methods can be integrated in different parts at the same time as most of them use a specific operator and an error metric to control the results. Two other categories have been added for most recent works: the appearance-preserving techniques which take into account attributes such as color, normal and texture coordinates and the out-of-core simplification methods which focus on the issue of large models using large amount of memory.

3.3.1.2 Manual preparation

The creation of various levels of details started in the flight simulator field. These LOD were made by hand by designers. But this kind of preparation became time-consuming and a difficult task. The necessity of automating this work appeared and thus the researches on mesh simplification as well. It appeared that the traditional refinement methods used to approximate curves and height fields has been abandoned for decimation methods in surface simplification algorithms.

3.3.2 Operators: how to reduce the number of polygons

Simplification operators can be local and then simplify the geometry and connectivity in a local region of the mesh, reducing the number of polygons or global and then operate over much larger regions and help to the simplification of the mesh topology.

3.3.2.1 Local

3.3.2.1.1 Face / edge swapping

The method used by **Hamann and Chen** [Hamann1994b] is based on a refinement for the simplification of tetrahedral convex complexes. Using a local least square polynomial approximation, weights are assigned to points. The most important points are inserted in the convex hull (boundary) of the original point set. When a point is inserted in the triangulation, a local simplification, a face swapping is performed in order to minimize the local approximation error.

3.3.2.1.2 *Vertex removal*

In order to remove redundant geometry information in the mesh, Schroeder et al [Schroeder1992] proposed a vertex decimation which uses the vertex removal operation. This operation consists in multiple passes over the vertices of a triangulated mesh. During a pass, a vertex is considered for removal by evaluating the decimation criteria. This vertex is deleted with all associated triangles and the resulting hole is retriangulated (see Figure 8). The algorithm then increases the threshold at which vertices are removed and makes another pass. The algorithm terminates when it reaches the desired simplification level specified as a number of triangles or in terms of the decimation criteria. The decimation criteria can be the distance of the vertex from the average plane of its surrounding triangles. While this method provides efficiency and quality, it is limited to manifold surfaces and it maintains the topology of the model.

See also: **[Renze1996], [Soucy1996], [Ciampalini1997]**

3.3.2.1.3 *Triangle removal*

The triangle removal operation consists in selecting a triangle for removal, selecting all triangles that are adjacent to the selected one and removing the selected triangles and finally retriangulating the region.

3.3.2.1.4 *Vertex pair collapsing / Virtual edge collapse / Vertex pair contraction*

The vertex pair collapsing (or virtual edge collapse) has been introduced by **Schroeder** [Schroeder1997]. A vertex pair operator collapses two unconnected vertices v1 and v2. Since these vertices do not share an edge, no triangle is removed but triangles surrounding v1 and v2 are updated as if an imaginary edge connecting v1 and v2 underwent an edge collapse (see Figure 8). It permits non manifold meshes and enable changes to the mesh topology such as closing holes and merging separate meshes. The difference with the edge collapse operator (described in the next section) is that the two vertices do not require to be merged by a node to share an edge.

See also: [Garland1997], [Popovic1997], [El-Sana1999a]

3.3.2.1.5 *Edge collapsing / Iterative edge contraction*

In 1993, **Hoppe** [Hoppe1993] introduced the edge collapsing operation a form of vertex merging. This operator collapses an edge (v_1,v_2) to a single vertex v and thus remove the edge (v_1,v_2) as well as the triangles spanning that edge. The inverse operation is called a *vertex split* and adds the edge (v_1,v_2) and the triangles adjacent to it. The edge collapsing simplifies the mesh while the vertex splitting adds detail to it. They are both represented in Figure 8. This operator is simple to implement but we must take care to not apply an edge collapse if it would cause a mesh foldover (see figure 7) or a topological inconsistency (a manifold mesh can become nonmanifold). The full-edge collapse, often abbreviated as simply edge collapse is a variant of edge collapse operator.

Here the collapsed vertex v may be a newly computed vertex. Hoppe takes into account only manifold surfaces; however this operator deals with non-manifold surfaces as well. It can close holes in the object but can not join unconnected regions.

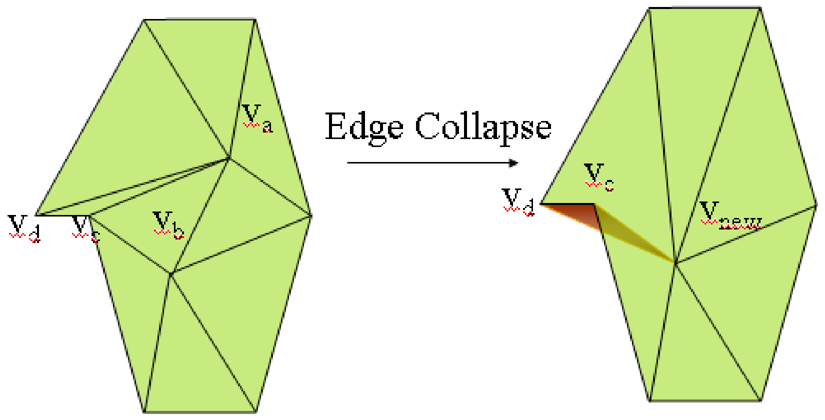


Figure 7: foldover issue[2]

See also: [Hoppe1996], [Geuziec1996], [Ronfard1996], [Algorri1996]

3.3.2.1.6 Half-edge collapsing

This operator, used by **Kobbelt** [Kobbelt1998], is a variant of edge-collapsing where the vertex to which the edge collapses to is one of its end points, that is $v_{new} = v_a$ or v_b. This operation is represented on Figure 8.

3.3.2.1.7 Triangle collapsing

Hamann [Hamann1994a] presented the triangle collapsing operator which collapses a triangle (v1, v2, v3) to a single vertex v. The edges that define the neighborhood of v are the union of edges of the vertices v1, v2, and v3. The vertex v obtained can be v1, v2, v3 or a newly computed vertex. A representation of this operator can be found in Figure 8. A triangle collapse is equivalent to tow edge collapses described earlier.

[2] A. Varsnhey. Level of Detail Management for 3D Games. Talkslides.

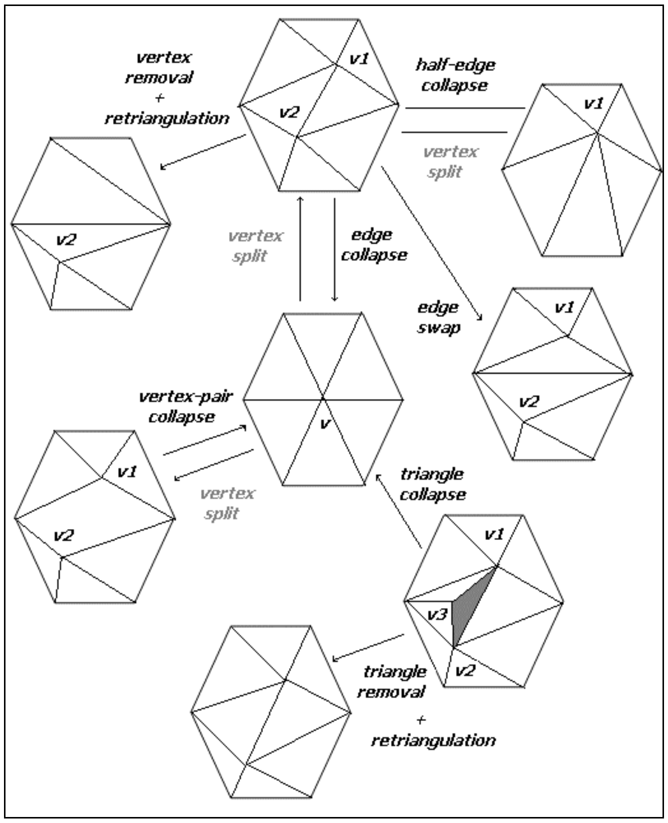


Figure 8: The different local connectivity operations

3.3.2.1.8 Cell collapse / Vertex clustering (form of vertex-pair contraction) / Vertex merging / Vertex collapsing

The vertex clustering starts by assigning a weight or importance to every vertex in the model. For example, **Rossignac and Borrel** [Rossignac1993] assigned higher importance to vertices attached to large faces, and to vertices in regions of high curvature (since such vertices are more likely to lie on the silhouette). This operator simplifies the input mesh by collapsing all the vertices contained in a certain volume (cell, cluster) to a single vertex (the most important vertex). Different approaches exist: the cell collapse can belong to a grid [Rossignac1993], or a spatial subdivision such as an octree [Luebke1997] or a volume in space [Low1997]. This simplification does not preserve the topology of the input mesh and the level of simplification and thus the quality of the resulting simplification depends on the resolution of the grid. This algorithm of Rossignac and Borrel deals robustly with messy models and degenerate meshes and is one of the fastest algorithms. However it presents less pleasing visually simplification results than those of slower algorithms and this simplification is sensitive to the orientation of the clustering grid since two identical objects at different orientations can produce quite different simplifications. To resolve this problem of uniform grid clustering, **Low and Tan** [Low1997] proposed an alternative to their algorithm using *floating-cell clustering.* They center a clustering cell on the most important vertex in the model and collapse all vertices within the cell to this most important vertex. In the process, delete all vertices in the cell from the list. And do again on the most important remaining vertex. They use spherical cells and thus they test the Euclidean distance (L2 norm) rather than the Manhattan distance (L1 norm) for cubical cell. The problem of Low and Tan is the asymptotic running time of their algorithm. While providing higher quality results, one drawback of this approach is that it requires sorting the vertices, which is generally an $O(n \log n)$ procedure, compared to the $O(n)$ running time for Rossignac and Borrel's original scheme.

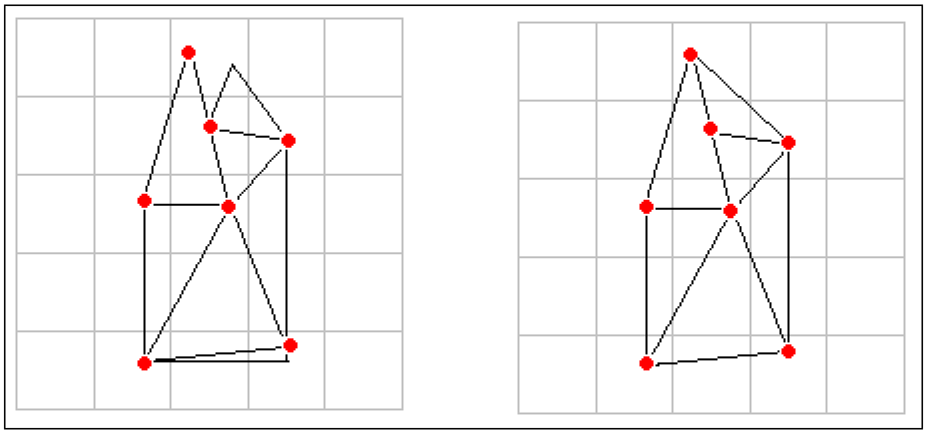


Figure 9: cell collapse or vertex clustering

See also: [Low1997], ([Luebke1997][Lindstrom2000a])

3.3.2.1.9 Polygonal merging / Face merging / Face clustering / Superfaces / Coplanar facet merging

Based on the same idea than vertex clustering but for faces, faces of the original mesh are grouped together in clusters. Then one face per cluster is selected and kept. Hinkler and Hansen [Hinkler1993] were the first one to propose merging nearly coplanar and adjacent polygons into larger polygons which are then triangulated. For example on Figure 8, the vertex removal could be compare to a merging of the five up faces into one that is triangulated in the final result of vertex removal operation. However polygon merging can merge together polygons and not only triangles.

See also: [Garland2001], [Kalvin1996]

3.3.2.2 Global

The global simplification operators modify the topology of the mesh in a controlled fashion. They are in general more complex then the local simplification operators.

3.3.2.2.1 Re-tiling / Re-meshing technique

Turk [Turk1992] distributes a new set of vertices over the surface of a model and connects them to create a retiling of a surface keeping fidelity of geometry and topology of the original surface. They found a way of using an estimate of surface curvature to distribute more new vertices at regions of higher curvature and noticed that a linear interpolation produced smooth transition between models that represent the same object at different LOD.

To summarize, the collapse operators (edge collapse, triangle collapse and cell collapse) are the simplest to implement. As edge, triangle or cell can be reduce to a single vertex in these approaches, they are well suited for implementing geomorphing between successive LOD. ***.*** *Fewer triangles need to be updated with the half-edge collapse than edge collapse which can make it more efficient. The cell collapse has the advantage to simplify both geometry and topology. It is a robust operator very simple to implement. Nevertheless this operator is not invariant with respect to rotation and translation and does not simplify the topology in a controlled fashion.*
The vertex removal is harder to implement than the collapse operators since it involves triangulating a hole in 3D.
The vertex removal operator and the edge collapse operator each have their strengths and weakness but neither is strictly better than the other. Each can produce triangle meshes than the other cannot.
Algorithms based on vertex removal and edge collapses have been able to obtain more drastic simplification (in terms of reducing the polygon count) and produce better looking simplifications.

3.3.3 Simplification process based on error or energy metrics

In summary, the similarity (or difference) of the shape have been described in a few features. They are shape differences, curvature differences and image differences. The curvature difference is simply measured by measuring differences in the curvature or the curvature is only used to decide which part should be preserved. Thus in this section, the geometrical shape difference measurements and image difference measuments are mainly described.

3.3.3.1 Vertex-vertex distance

The simplest approach to measure the error of a simplified model is to measure the distance between the original vertices and the simplified vertices. The main issues here are to find the accurate correspondences between vertices and the appropriate simplification operator. One main problem with this measure approach is that even if vertices did not change, the surface could have changed dramatically. We can obtain a Hausdorff distance equal to 0 while the distance at the interior points of the surface can be much greater. Nevertheless, this distance between vertices can compute a useful bound on the distance between surfaces in certain circumstances such as vertex merging operations (edge collapse, cell collapse, etc.). We can quote some works based on this distance error measure:

- the uniform grid-based vertex clustering algorithms of **Rossignac and Borrel** [Rossignac1993] and **Cohen** [1998],
- the hierarchical grid-based vertex clustering algorithm of **Luebke and Erikson** [Luebke1997], they evaluate the error represented by a node according to the bounding sphere containing the node's region of support, reasoning that vertices cannot move further than the diameter of the bounding sphere. They then unfold node so as to minimize the screen-space error obtained by projecting this sphere onto the screen. "Vertices on the screen can move as far as x pixels from their original position. Minimize x without exceeding the triangle budget."
- the floating cell-based vertex clustering algorithm of **Low and Tan** [Low1997]

This method is appropriate when the topology is changing.

3.3.3.2 Vertex-plane distance

Given a plane with a unit normal n and signed distance from the origin D the shortest distance from point p = (x,y,z) to the plane is:

$$d = n \cdot p + D = n_x x + n_y y + n_z z + D$$

This distance is more computationally efficient than the distance between two points. The models used here are composed of planar polygons rather than infinite planes. Thus the vertex-plane distance methods do not really provide a bound on the maximum or average distance between models. This error measure requires in general another metric or a tool such as Metro. Simplification methods based on vertex-plane distance work well in practice, they are fast and moderately accurate, tending to produce LODs with measurably low error for a given polygon count. Following are described two important algorithms using this distance measure: the maximum supporting plane distance and the quadric error metric.

3.3.3.2.1 Maximum supporting plane distance

Ronfard + Rossignac [Ronfard1996] use edge collapse operations, measuring for each potential edge collapse the maximum distance between the simplified vertex and each of its supporting planes. They store plane sets and compute the max distance.

3.3.3.2.2 QEM, Quadric Error Metric by

Garland and Heckbert [Garland1997] modified the error metric of Ronfard and Rossignac explained above to make it faster and more compact in storage space. The quadric error metric replaces the maximum of squared vertex-plane distances with the sum of squared vertex-plane distances as:

$$E_v = \sum_{p \in planes(v)} (p \cdot v)^2 = \sum_{p \in planes(v)} (v^T p)(p^T v) = v^T \left[\sum_{p \in planes(v)} pp^T \right] v = v^T \sum_{p \in planes(v)} Q_p v = v^T Q_v v$$

The quadratic form or error quadric Q_p is a 4x4 symmetric matrix, computed as pp^T and represented using 10 unique floating point numbers. They store the quadratic form and compute the sum of square distances.

The quadratic error metrics algorithm offers a combination of speed robustness and fidelity not previously seen. It is a relatively simple algorithm to implement. It measures the cumulative surface distortion introduced by a series of vertex merge operations. It iteratively merges pairs of vertices until it reaches the desired level of simplification. It begins by finding candidate vertex pairs, including all vertices that share an edge. The candidate vertex pairs are entered into a priority queue sorted by their quadric error, and merged in order of increasing resulting error. Merging a vertex pair changes the error associated with other pairs in the local mesh neighborhood. These pairs are reevaluated and reinserted into the priority queue. Simplification terminates when the desired number of polygons is reached.

A quadric is a 4x4 symmetric matrix that captures information about one or more planes. Given a vertex coordinate v and a quadric Q representing a set of planes, evaluating (v^TQv) gives the sum of the squared distances from the vertex to each plane. Quadrics are additive, thus the quadric of the union of two sets of planes is the sum of the quadrics of each set. When two vertices vi and vj with quadrics Qi and Qj are merged to form a new vertex vk, the quadric Qk is Qi+Qj. The error associated with this merge operation is ($v_k^TQ_kv_k$) which represents the sum of the distances from vk to all the planes accumulated in both Qi and Qj. The quadric error metrics algorithm begins by calculating the fundamental error quadric for each triangle in the original model, and initializing the quadric of each vertex in the original model to the sum of the quadrics of its associated triangles. The error at each initial vertex is 0 since the vertex lies at the intersection of the planes of its triangles.

3.3.3.3 Vertex-surface distance

This vertex-surface distance consists in mapping point set to the closest points of simplified surface and computing the sum of square distances. It attempts to map one surface to another. The vertices of the original model are mapped to their closest points on the polygons of the simplified surface. Vertex-surface approaches are generally slower than vertex-plane approaches because suitable mappings must be found for the many input vertices. The well-known algorithms of Hoppe using this distance measure are the progressive meshes and the mesh optimization described below.

3.3.3.3.1 Energy function approaches

3.3.3.3.1.1 The mesh optimization

Hoppe et al. [Hoppe1993] proposed to resolve the mesh optimization problem, i.e. to produce a mesh of the same topological type as an initial triangular mesh that fits a set of data points well and has a small number of vertices. To obtain this result, they minimize the following energy function:

$$E(M) = E_{dist}(K,V) + E_{rep}(K) + E_{spring}(K,V)$$

Where: K is a simplicial complex

V is a set of vertex position which defines a mesh M = (K,V) that minimizes the above energy function.

Edist is the sum of squared distances from the points X = {x1,…,xn}. It is a distance energy that measures the closeness of fit.

Erep is a representation energy that penalizes meshes with a large number of vertices. It is set to be proportional to the number of vertices m of K.

Espring is a regularizing term that places on each edge of the mesh a spring of rest length zero and spring constant k.

This method can recover sharp edges and corners.

3.3.3.3.1.2 The progressive meshes

Hoppe [Hoppe1996] provides thanks to his algorithm, a continuous spectrum of detail (a continuous LOD), stored as a simple base mesh and a series of refinement operations. This method follows a greedy edge collapsing algorithm, minimizing the change to his energy function with each successive edge collapse. During the progressive meshes process, edge collapses are prioritized according to how they affect the following energy function:

3.3.3.3.2 Metro tool

The Metro tool described in section II to measure the output quality of simplification algorithms of **Cignoni et al.** [Cignoni1998] uses a point-surface distance metric.

3.3.3.4 Surface-surface distance

They bound the maximum distance between input and simplified surface. A surface-surface distance metric can provide the strongest guaranteed bounds on the error of a simplified surface. This metric considers all points on both the original and simplified surface to determine the error at a given stage of the simplification process. This approach can be really useful in medicine and scientific visualization applications, and in the case of scanned models. Indeed, it can be necessary to minimize the error from this entire reconstructed surface during the simplification process. We will describe some main algorithms using this approach.

3.3.3.4.1 Simplification envelopes

This method can also be classified as a global decimation approach. **Cohen and Varshney** [Cohen1996] start by constructing inner and outer envelope surfaces which together enclosed the original surface. The original vertices are displaced by a distance ≤ε along their normal vectors checking that self-intersection with the envelope polygons does not happen. The inputs of this algorithm are a manifold triangle mesh and an error tolerance, while the output is a simplified mesh with a maximum error that is as closed as possible to the tolerance without exceeding it. Each original vertex is inserted in a queue for a vertex removal operation. The resulting holes are filled using a greedy triangle insertion. If a triangle intersects an envelope surface or the rest of the simplified surface it can be not used to fill the hole. The multiple discrete LODs are found by simplifying the original model several times with different error thresholds or by cascading the simplifications (the 1st LOD is simplified to produce the 2nd one, the 2nd to produce the 3rd one and so on).

3.3.3.4.2 Preserving volume technique

Gueziec [Gueziec1996] uses a bounding volume approach to measuring simplification error. His error volumes are measured locally and grown iteratively. This algorithm uses a priority queue of edge collapses ranked by maximum error. A sphere with zero radius is initially centered at each vertex. As the model is simplified the spheres of the simplified mesh vertices grow at different rates to indicate the maximum local error. This error is defined across the entire surface of the mesh by interpolating the vertex sphere radii across each triangle. The union of these varying radius spheres swept over the entire triangle mesh forms an error volume and the radius at any given point of the mesh bounds the maximum geometric deviation error at that point. This error sphere approach can also be used to measure and optimize maximum errors in color, texture coordinate and normal attributes.

3.3.3.4.3 Plane mapping

As for the Simplification Envelopes algorithm, the Plane Mapping approach of **Bajaj and Schikore** [Bajaj1996] is based on vertex removal operations. The difference comes from the measure of the error that uses the maximum pointwise mapping distance, where the mapping function is locally defined by orthogonally projecting the affected set of triangles before and after the vertex removal operation onto the same plane. Points of each surface that project to the same location on the plane are corresponding point pairs. In other words the initial mesh is superposed on the resulting mesh after vertex removal and retriangulation and the edge crossing points are chosen for error measurement.

3.3.3.4.4 Hausdorff approach

Klein, Leibich and Straβer [Klein1996] described an algorithm based on vertex removal and the Hausdorff distance. They ensure that for each point in the original mesh there is a point in the simplified mesh with a Hausdorff distance smaller than a user-defined error tolerance.

3.3.3.4.5 Appearance-preserving simplification

Since this method of **Cohen et al.** [Cohen1998] handles appearance attributes such as color, normal and texture coordinates, we refer you to the section E on appearance preserving for its description.

3.3.3.5 Image metric

In 2000, **Lindstrom and Turk** [Lindstrom2000b] use a geometry driven approach to place new vertices for edge collapses using their memoryless algorithm but use an image-space error metric to prioritize the edge collapses in the queue. They measure the error by rendering multiple images of the object using a sphere of virtual cameras. The cameras are placed at the 20 vertices of a dodecahedron. Each of these 20 camera viewpoints renders an image of the original model and of the simplified model. A single light source is positioned near each of viewpoints so that the model is fully illuminated in each image. The human vision relies primarily on luminance for detection and recognition of form and texture. A pixel is typically represented by a color and a luminance. Thus a root-mean-squared error of pixel luminance values is computed between the two sets of pixels from all the cameras. The difference between the pixels of the two m×n pixels images is then determined by the difference of luminance (Y^0 and Y^1) between these two, by calculating the following RMS difference:

$$d_{RMS}(Y^0, Y^1) = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(y_{ij}^0 - y_{ij}^1)^2}$$

This RMS error is the edge's key in the priority queue. This method incorporates visual errors due to a number of sources, such as motion of the silhouette and deviation of color, normal and texture coordinate attributes. The number of viewpoints clearly involves a trade-off between simplification speed and the risk of missing an important effect. Nevertheless the algorithm is slower than the slowest geometric algorithms, since rendering and rerendering multiple images for every edge collapse is an expensive way to measure error. Image-driven solves the problem of how to weight appearance attributes such as normals, color, and texture versus geometric distortion of the surface. The advantages of this approach are the excellent fidelity, the good silhouette preservation, the drastic simplification of hidden portions of the model and texture-content-sensitive simplification. Nevertheless image-driven simplification can be classified as the slowest modern algorithm for producing LODs and is very slow for very large models.

3.3.4 Hierarchical representation / Simplified model: *how to represent polygons*

3.3.4.1 Grid sub-sampling

A grid sub-sampling is a uniform subdivision that consists in subdividing polygons with a regular grid. In 1994, **Kumler** [Kumler1994] compared regular grids and general triangulation. He affirmed that general triangulations require three to ten times the memory of regular grids with same number of vertices.

3.3.4.2 Hierarchical subdivision

A common way to represent a polygon mesh is to have a list of vertices and a list of faces storing pointers for its vertices. This representation can become inconvenient in the domain of mesh simplification, which often requires collapsing an edge into a single vertex for example. For this reason, researchers needed to use some methods to subdivide these mesh in a certain way to facilitate the management of vertices and faces.

3.3.4.2.1 Basic subdivisions

Quadtrees (2D) and Octrees (3D) refine resolution of voxels hierarchically. We describe them below.

3.3.4.2.1.1 Quadtrees -> 2D pyramidal

In his paper Samet [Samet1984] described quadtrees as based on the recursive decomposition of space. Quadtrees can have different type of data represented such as regions, vertex, etc. For example, the most studied approach has been the *region quadtree* based on the successive subdivision of the image into four equal-sized quadrants. Then the leaf node of the tree corresponds to the region for which no further subdivision is necessary. Thus Von Herzen [VonHerzen1987] triangulate deformed intersecting parametric surfaces using surface quadtrees. He chose an adaptive sampling and restricted rather than unrestricted quadtrees. The drawback of adaptive sampling (more efficient than uniform subdivision) is the recursive subdivision process and the set of subdivision criteria (such as curvature, intersection, proximity and silhouette).

3.3.4.2.1.2 Octrees ->3D pyramidal

Andujar [Andujar1996] worked on an automatic simplification of general 2D manifold polyhedra using an intermediate octree representation. Later Luebke and Erikson [Luebke1997] uses the cell collapse (or vertex clustering) operator with a recursive octree providing an adaptive version of the Rossignac-Borrel uniform grid approach. They use an octree vertex hierarchy that they called *vertex tree* in which every node has at most eight children. A cell collapse operation merges all vertices in an octree cell to the single "most important" vertex so that interior nodes as well as leaves represent vertices of the original model. Using a screen space error threshold, they collapse nodes which occupy a small amount of the screen.

3.3.4.2.2 Triangulations

3.3.4.2.2.1 Quaterny triangulation

Gomez and Guzman [Gomez1979] approximated height fields thanks to their quaterny triangulation. In their method, each triangle is subdivided into four sub-triangles until a maximum error tolerance is met.

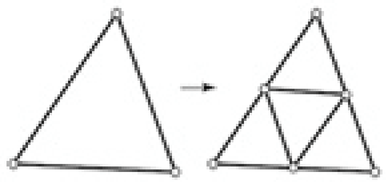


Figure 10: Quaterny triangulation

3.3.4.2.2.2 Ternary triangulation

DeFloriani et al. [DeFloriani1984] introduced a hierarchical ternary triangulation method in which points are inserted in triangle interiors and each triangle is split into three subtriangles by adding edges to its vertices.

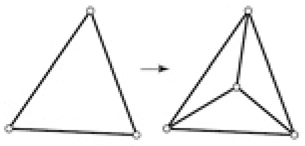


Figure 11: Ternary triangulation

3.3.4.2.3 Voxelizing

Voxelizing consists in a partition of space into uniform grid where grid cells are voxels. The color and the density are stored in each voxel. **He et al.** [He1995][He1996] and **Nooruddin and Turk** [Nooruddin1999] based their simplification of the topology on a voxelizing or conversion of the input objects into a volumetric grid. Then they apply a topology simplification operation in volumetric domain and finally use an isosurface extraction method to convert the volumetric densities into a triangle mesh. While He uses a low-pass filtering as topology simplification operator, Nooruddin uses morphological operations of dilatation and erosion. He et al. place a filter at each grid value and the approach computes the intersection of the geometric primitive with the filter kernel extent. This intersection amount produces a filtered density value for each voxel, enabling data sets derived from polygonal meshes and implicit functions to be treated in the same manner as innate volumetric data sets such as those from CT or MRI scans. Finally they apply a low-pass filter to each of the grid values of the volumetric buffer and use an isosurface reconstruction using a method such as the Marching Cubes algorithm **[Lorensen1987**].

3.3.4.2.4 Wavelet

In order to represent some restricted meshes with subdivision connectivity, **Lounsbery et al.** [Lounsbery1994] introduced a multiresolution representation. Since a simple base mesh (the complex object with a low resolution), they progressively apply local corrections called wavelet coefficients. They add smaller wavelet coefficient as the viewer approaches the object and remove them as the viewer recedes. Threshold for removal can be chosen such that the resulting approximation is guaranteed to be within a specified error tolerance of the original mesh. Nevertheless the subdivision connectivity restriction stays an unresolved issue with this method. Thus **Eck et al.** [Eck1995] proposed a method converting completely arbitrary meshes into multiresolution form using these wavelet coefficients. To avoid the adaptive triangulation issue, **Gross et al.** [Gross1996] used the wavelet transform (WT) to control the data approximation. They used inverse WT as criteria for vertex removal and introduced a new type of wavelet space filtering to help defining local regions of interest. They build a quadtree representation of the initial mesh by removing dyadic vertices. Wavelet coefficients after 3D wavelet transform are grouped together in a tree.

3.3.4.3 Vertex/Point tree hierarchy

Aiming to perform real-time adaptive simplifications, **Xia and Varshney** [Xia1996] built a binary vertex hierarchy structure over the vertices of the model that they called a *merge tree*. This method consists in storing edges collapses in a hierarchical manner. The construction of this merge tree is done off-line as a preprocessing stage before the interactive visualization. They use *ecol/vsplit* transformations to create a simplification hierarchy. First all vertices are entered as leaves at level 0 of the tree. Then, for each level, a set of ecol transformations is selected to merge pairs of vertices and the resulting subset of vertices is promoted to level l+1. These ecol are chosen based on edge lengths and in order that their neighborhoods do not overlap. The topmost level of the tree (the *forest*) corresponds to the vertices of a coarse mesh. The binary hierarchy structure has the advantage that individual nodes require less storage since for example nodes need not store the number of children. Plus, a node can avoid storing indices of both children by guaranteeing that sibling nodes are stored in adjacent positions of an array. Lending to efficient traversal this regular structure of a binary hierarchy has also the disadvantage of very fine granularity it imposes. Since each node represents an edge collapse folding a node removes only 2 triangles from the mesh. Thus the hierarchy will contain many nodes and many folds and unfolds will be needed each frame to achieve the desired LOD. **Hoppe** [Hoppe1997] based his method of progressive meshes on this vertex hierarchy processing, but let the hierarchy be formed by an unconstrained, geometrically optimized sequence of vsplit transformations (from an arbitrary PM) and to introduce as few dependencies as possible between these transformations, in order to minimize the complexity of approximating meshes.

3.3.4.4 Polygonal hierarchy

In 1989, De Floriani [DeFloriani1989] created a *Delaunay pyramid* i.e. a hierarchy of Delaunay triangulations. Each triangle stores the set on input points it contains and the error of its candidate point. On each pass, the set of triangles is scanned to find the candidate if highest error, this point is inserted using incremental Delaunay triangulation and the candidates of all the triangles in the modified region are recomputed. Recomputing the candidate of a triangle requires calculating the error at each point in the triangle's point set. Cignoni et al. [Cignoni1994] extended the 2D Delaunay pyramid of De Floriani for representing terrain data in 3D. They proposed a Delaunay refinement strategy where the choice of the vertex to select is based on the point causing the largest error with respect to the original scalar field. Their multiresolution model is based on a decomposition of the three-dimensional domain into tetrahedra.

3.3.4.5 Sphere hierarchy

Kim et al. [Kim1998] proposed a new data structure called multiresolution view sphere that they called later AGSphere [Kim2004]. The sphere is then used to map the directional relationship between object surface and the viewpoint. Their goal was to implement a fast algorithm capable to render object while preserving its silhouette. Using quad-like tree and half-edge structure, they developed an efficient mesh merging algorithm based on a cell hierarchy.

3.3.4.6 Simplicial complex hierarchy

Popovic and Hoppe [Popovic1997] introduced a new format for storing and transmitting triangulated models: the progressive simplicial complexe (PSC). It is a generalization of the progressive meshes (PM) that start with a coarse base model (a single vertex) that they progressively ameliorate with refinement transformations. The difference is that here the model can be an arbitrary triangulation (any dimension, non-manifold, non-orientable, non-regular) and it allows topological changes. Popovic and Hoppe use a different refinement transformation, the generalized vertex split that encodes both the geometry and topology of the model. Their algorithm allows geomorph between any pair of models in the sequence. DeFloriani et al. [DeFloriani2004] treated non-regular, non-manifold two-dimensional simplicial meshes that they call *triangle-segment* meshes at different LODs. They developed a model for multiresolution representation of such triangle meshes that they called Non-Manifold Muli-Tesselation (NMT). And they constructed a compact data structure for triangle-segment meshes through vertex-pair contractions. This structure represents both connectivity and adjacency information with a small memory overhead.

3.3.5 Appearance preserving methods

Quite recently, researchers take into account not only the geometry but the appearance attributes of the mesh to simplify for a better fidelity. Researchers tried to preserve attributes of colors, normals and texture coordinates, while other focused their work on the parameterization of the mesh for better smoothness and which can be used for texture mapping.

3.3.5.1 General appearance preserving

Instead of taking into account only the geometry of the simplified meshes, some researchers are interested by the appearance attributes such as color, normal and texture coordinates. Analysing surface geometry (shape) and color separately became necessary. Thus, **Certain et al.** [Certain1996] extended the work of Eck based on the multiresolution analysis to simplify arbitrary meshes [Eck1995] to colored meshes by separately analyzing shape and color. They make use of texture mapping hardware to render the color at full resolution. **Hoppe** [Hoppe1999] proposed as well to separate the geometric error from the attribute error. Based on the work of Garland and Heckbert on quadric error metric [Garland1997], he developed an improved quadric error metric for simplifying meshes with attributes. His method incorporating attributes in quadrics, requires less storage and computation and produces better-looking results. Researchers realized the necessity to find new accurate metrics for attributes of the mesh. Thus, **Cohen et al** [Cohen1998] sampled the surface position (represented by the coordinates of the polygon vertices), the surface curvature (represented by a field of normal vectors across the polygon) and color attributes (represented as a field across the polygons) and introduced a new metric called "texture deviation metric". The user-specified error tolerance ε is a screen-space deviation in pixels units. A particular point on the surface with some color and some normal may appear to shift by at most ε pixels on the screen. The primary visual artifact is then a screen-space distortion of the texture and normal maps, which the user can control by choosing an acceptable tolerance for the texture deviation at run-time. The texture deviation between a simplified mesh Mi and the original mesh Mn at a point pi€Mi is defined as ||pi-pn|| where pn is the point on Mn with the same parametric location in the texture domain. Cohen et al track texture deviation conservatively by storing a bounding error box at each mesh vertex. Possible input models of their algorithm are CAD models with per-vertex normals and a single color, radiositized models with per-vertex colors and no normals, scientific visualization models with per-vertex normals and per-vertex colors, and textured models with texture-mapped colors with or without per-vertex normals. **Erikson and Manocha** [Erikson1999] treat models that contain both non-manifold geometry and surface attributes, such as radiositized scenes, textures and scanned meshes, CAD environments and terrain models. They grow error volumes for appearance attributes as well as geometry. Their General and Automatic Polygonal Simplification (GAPS) uses an adaptive distance threshold and a surface area preservation along with a quadric error metric to join unconnected regions of an object. In the opposite, **Reddy** [Reddy2001] is the first researcher to have based his researches of LOD system on visual perception. He proposed a visualization system that specifies how much detail a human can perceive under various circumstances by removing detected imperceptible details in an image.

3.3.5.2 Smooth parameterization /Parameterization for smoothing

In order to ameliorate the fidelity of the simplified models, researchers are interested by their smoothing and used parameterization to obtain good results. Parameterization is the process of mapping a surface onto regions of the plane. It allows operations on a surface to be performed as if it is flat. It is essential for texture mapping, morphing, re-meshing and surface reconstruction. The mapping of curved surface can be *authalic* (area-preserving) or *conformal* (angle-preserving) but never *isometric* (distance-preserving, i.e. authalic and conformal at the same time). Parameterization can be done after the mesh simplification or during the process of simplification.

Local parameterization

Most of conformal parameterization only deal with genus zero surfaces and have to segment the surfaces into patches. These methods decompose meshes into topological disks and then parameterize each patch individually. Thus, after the mesh optimization of **Hoppe et al.** [Hoppe1993], who proposed an approach of energy function minimization, **Eck et al** [Eck1995] introduced discrete conformal parameterization in 1995. They formulated the parameterization problem as the minimizer of an energy which measures the quality of the parameterization. They tile an arbitrary topology surface in patches or charts (regions each homeomorphic to a disc) grouped in an “atlas of charts” using a Voronoi-like decomposition and Delaunay triangulation and iteratively improves local parameterization by cycling over sub-regions of the mesh. The conformality of the map consists in scaling the metric and allows the preservation of angle-preserving property. This local parameterization is based on the theory of harmonic maps to minimize the distortion. Find a parameterization consists here in fixing a homeomorphism g between the boundaries of D (topological disk DCM M the original mesh) and the boundary of the polygonal region P (C$\mathbf{R}^2$) and then there is a unique harmonic map h: D -> P that agrees with g on the boundary of D and minimizes metric dispersion, i.e. minimizes the extent to which a map stretches regions of small diameter in D which is equivalent to the measure of metric distortion. Similarly, **Levy** [Levy2002] uses conformal parameterization using as well texture atlas. Their parameterization is based on a least-square approximation of the Cauchy-Rieman equations. The drawback of an atlas is the presence of visible seams on the surface.

Global parameterization

The previous local conformal parameterization methods introduce discontinuities along patch boundaries and conformality can not be preserved everywhere. For these reasons remeshers used global conformal parameterization. **Khodakovsky et al** [Khodakovski2003] proposed a new parameterization algorithm for arbitrary topology surface meshes: a globally smooth parameterization with low distortion. The parameterization can be classified according to their rate distortion (r/d) performance. Firstly, Khodakovski et al contruct the base complex which encodes the topological relation between patches. This base complex governs the patch layout and forms the parametric domain of the surface. Then the control the smoothness and distortion of the mesh in particular at patch boundaries. Khodakovsky et al use the vertex removal operator and retriangulate the hole left by the removed vertex and its incident triangles that admits little distortion. They use the mean-square magnitudes relative to the bounding box diagonal and the L2 error for their error metrics. But contrary to Levy [Levy2002], the algorithm of Khodakovsly et al does not fix the patch boundaries. In an alternative approach based on the control of smoothness and distortion, **Gu and Yau** [Gu2003] proposed a global conformal smooth parameterization for surfaces with arbitrary topologies, with or without boundaries. They were the first to solve the problem of global conformal parameterization of nonzero genus surfaces with boundaries. Their method is based on the properties of gradient fields of conformal maps which are closeness, harmony, conjugacy, duality and symmetry.

Spherical parameterization

Gotsman, Gu and Sheffer [Gotsman2003] used a different approach for their parameterization. As a closed manifold genus zero mesh is topologically equivalent to a sphere, they parameterize a triangle mesh onto the sphere, assigning a 3D position on the unit sphere to each of the mesh vertices, minimizing the distortion measure, and avoiding that individual planar triangle overlap. **Praun and Hoppe** [Praun2003] shows that geometric models described by closed genus zero surfaces are better parameterized by sphere since it does not require cutting the surface into disks. They resample spherical parameterization into geometry images (see next section).

3.3.5.2.1 Geometry images GIM

Gu et al [Gu2002] proposed to remesh an arbitrary surface onto a completely regular structure called geometry image. They capture geometry as a simple 2D array of quantized points. Surface signals such as normals and colors are stored in similar 2D arrays using the same implicit surface parameterization. Texture coordinates are absent. They create geometry images by cutting an arbitrary mesh along a network of edge paths and parameterize the resulting single chart onto a square. Geometry images can be encoded using traditional image compression such as wavelet-based codes. Nevertheless this technique can not represent non manifold geometry. As error metric they use a Peak Signal to Noise Ratio PSNR = 20 $\log_{10}$ (peak/d) with peak the bounding box diagonal, and d the symmetric RMS Hausdorff error (the geometric distance) between the original mesh and the reconstructed geometry. Unfortunately models of high genus can be problematic and may require long cut to open up all the topological handles.

3.3.5.3 Texture mapping

Maillot et al [Maillot1993] used an atlas approach for texture mapping. They partition a mesh into charts based on bucketing of face normals. Their parameterization method optimizes edge springs of non-zero rest length. Based on iterative edge contraction and quadric error metrics, **Garland and Heckbert** [Garland1998], Garland and Heckbert extend their previous algorithm on quadric error metric [Garland1997] to handle vertex attributes. **Sander et al** [Sander2001] have exposed their method that constructs a progressive mesh for an arbitrary mesh where all meshes in the progressive mesh sequence share a common texture parameterization. They take into account closed objects by splitting a closed object into a series of patches called “atlas of charts”. This texture atlas allows the same texture image(s) to be used for all LOD mesh approximations in the progressive meshes sequence. Sander et al need to study the trade-off between texture quality (stretch) and geometric quality (deviation). In order to resolve the problem of texture boundary (see Figure 12), **Kim and Wohn** [Kim2001] proposed a multiresolution texture mapping method for AGSphere and vertex-tree based structures. They generate a set of hierarchical texture images to be mapped onto the hierarchical geometry. The error metric used is similar to the one introduced by Cohen [Cohen1998]: the texture deviation metric.

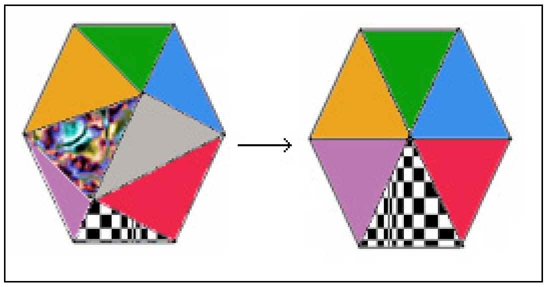


Figure 12: Texture boundary issue during an edge collapsing

3.3.6 Out-of-core simplification methods OoCS (for very large arbitrarily models)

Face to the explosion of the models size the last few years, researchers had to find methods for out-of-score visualization, i.e. for models that are too large to fit in main memory. For example, the Michelangelo's sculptures made by IBM [Bernardini2002] and Stanford University [Levoy2000] contain up to two billions triangles. These enormous datasets became a challenge not only for the rendering or compression part but also for the simplification methods. The "memoryless" approach of **Lindstrom and Turk** [Lindstrom1998] minimizes the use of memory and is used for large models. They preserve volume and area as simplification progresses using an incremental rather than total error. It is one of the fastest and most memory efficient algorithm but it is not enough for these really large datasets. The internal storage to represent a n-vertex model rises to a minimum of 160n bytes with this method. Thus simplifying a one billion vertex model to a few million triangles would require 160GB of RAM and weeks to compute. Thus researchers proposed out-of-core simplification algorithms based on spatial clustering, surface segmentation or streaming.

3.3.6.1 Spatial clustering

Lindstrom [Lindstrom2000a] improved in execution time and quality the vertex clustering algorithm of Rossignac and Borrel [Rossignac1993] by using the quadric error metric introduced by Garland and Heckbert [Garland1997] which has been later improved by Lindstrom and Turk [Lindstrom1998] for positioning vertices. His algorithm has the advantage of computing only one single pass over the input mesh and not using any disk space beyond the input mesh. Nevertheless, mesh of arbitrary large size can be simplified but the complexity (the number of vertices or triangles) of the output mesh is limited by the amount of memory available. The same problem is present in the extension of this algorithm by **Shaffer and Garland** [Shaffer2001] who makes two passes over the input mesh. Their algorithm requires more RAM than OoCS in order to maintain a BSP-tree and additional quadric information in-core. To avoid this kind of issue, **Lindstrom with Silva** [Lindstrom2001], extended this previous algorithm by making it independent of the available memory of the host computer. Their technique uses disk instead of main memory but avoiding costly random accesses. They also focused on the preservation of surface boundaries without making use of connectivity information and they constrained the position of the representative vertex of a grid cell to an optimal position inside the cell. Later **Garland and Shaffer** [Garland2002] proposed a combined approach with two phases both based on quadric error metrics. The first phase consists in an out-of-core uniform one-pass clustering while the second phase applies an in-core iterative edge contraction to produce approximation of good quality.

3.3.6.2 Surface segmentation

Bernardini [Bernardini1999] uses a radically different approach to out-of-core simplification by segmented the model up into patches that can be simplified independently in-core and then merged in a preprocessing phase. **Hoppe** [Hoppe1998] uses a similar method for creating hierarchical LOD for height-fields. His technique has been generalized to arbitrary meshes by **Prince** [Prince2000]. Based on a similar surface segmentation, **El-Sana and Chiang** [El-Sana2000] proposed an external-memory algorithm to support view-dependent simplification of datasets that do not fit in main memory. During the run-time only the portions of the view-dependence tree that are necessary to render the given LOD are kept in main memory. Still to resolve the problem of RAM size, **Cignoni et al** [Cignoni2003] presented a data structure called Octree-based External Memory Mesh (OEMM) which supports external memory management of huge meshes. Besides, in order to enhance the quality of the output from the simplification, **Choudhury and Watson** [Choudhury2002] proposed a fully adaptive simplification of massive meshes at reasonable speed.

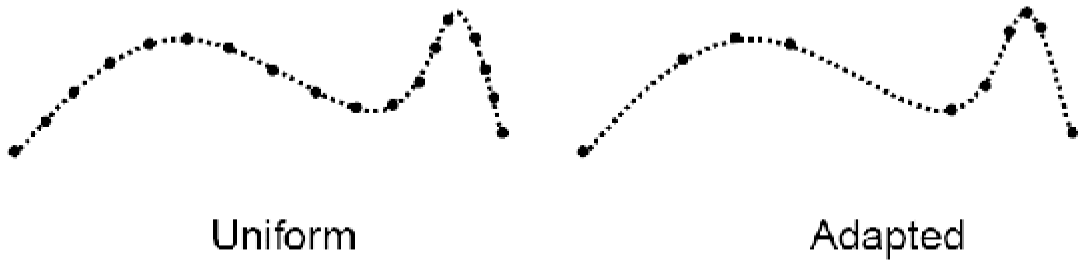


Figure 13: Difference between adaptive and uniform

3.3.6.3 Streaming

Wu and Kobbelt [Wu2003] proposed an out-of-core mesh decimation algorithm that takes in consideration input and output of arbitrary size. They read the input from a data stream in a single pass and write the output to another stream while using only a fixed-sized in-core buffer. They use then an incremental mesh decimation based on edge collapses and the quadric error metric. Recently, **Isenburg et al.** [Isenburg2003] introduced the concept of "processing sequence" which fixes a traversal order of the mesh and supports access to full connectivity and geometry information for the active elements of this traversal. They proved that this concept can be useful in other areas such as parameterization, remeshing or smoothing for which only in-core solutions exist. This representation allows streaming very large meshes through main memory while maintaining information about the visitation status of edges and vertices. This processing sequence provides an ideal input to Wu and Kobbelt's stream-based method.

3.3.7 Summary

In the next table, we tried to group together the works which seem to us the determinant results in the simplification mesh field. And we tried to compare these methods showing their main features to take into consideration.

Simplification methods	*Topo.*	*Inc.*	*App.*	*View*	*Surfaces*	*Global/Local operator*	*Approx.*	*QEM*
Operators								
Schroder et al 1992 *	Yes	Yes	No	Indep.	M	Vert. rem.	Vert-plane dist.	No
Rossignac Borrel 1993	No	No	No	Indep.	NM	Vert. clust.	Hausdorff dist.	No
Hoppe 1996	Yes	Yes	Yes	Indep.	O 2-M	Edge coll.	Distance energy	No
Hoppe DeRose 1993	Yes	Yes	No	Indep.	M	Global	Distance energy	No
Cohen Varshney 1996*	Yes	No	No	Indep.	O M	Global+Local	$L\infty$	No
Garland Heckbert 1997 *	No	Yes	No	Indep.	NM	Edge coll.	L2	Yes
Hierarchy								
DeFloriani 2004 *	Yes	Yes	No	Indep.	NM	Vert. pair coll.	None	No
Luebke Erikson 1997	No	No	No	Dep.	M+NM+CAD	Vert. clust.	Screen-space	No
Hoppe 1997	Yes	Yes	No	Dep.	M	Coll./Split.	Screen-space	No
Xia Varshney 1996	Yes	Yes	No	Dep.	M	Edge coll.	Euclidean dist.	No
Popovic Hoppe 1997	No	Yes	No	Dep.	NO,NR 2-NM	Vert. pair coll.	Distance energy	No
Kim 2004	Yes	No	Yes	Dep.	M	Vert. rem.	L2	No
Appearance preserving								
Cohen 1998	Yes	Yes	Yes	Dep.	CAD	Edge coll.	Texture deviation	No
Lindstrom 2000b	No	Yes	Yes	Dep.	M	Edge coll.	RMS	No
Reddy 2001*	Yes	Yes	No	Dep.	H	Ref.	Screen-space	No
Eck DeRose 1995	Yes	Yes	Yes	Indep.	M	Delaunay ref.	$L\infty$ + Dist. energy	No
Gu 2002	Yes	Yes	Yes	Indep.	2-M	Triang. rem./Ver.rem	RMS Hausdorff	No
Sander 2001	No	Yes	Yes	Indep.	M	Edge coll.	L2 + Linf	No
Kim 2001	Yes	Yes	Yes	Indep.	M	Loc. reg.+Vert.rem.+Edge coll.	Texture deviation	No
Out-of-core								
Lindstrom 2000a	No	Yes	No	Indep.	M	Vert. clust.	L2 + Hausdorff	Yes

LEGEND:

Topo.: Yes if topology is preserved, No otherwise
Inc.: Yes if incremental, No otherwise
App.: Yes if appearance is preserving, No otherwise
View: Indep. if view-independent, Dep. if view-dependent
Surfaces: treated surfaces types with:
  M: manifold,
  NM: non-manifold,
  H: Height fields,
  CAD: Computer-Aided Design models,
  NO: non-orientable,
  NR: non-regular,
  O: orientable)
Global/Local operator: Global or local operators can be:

Local refinement:	**Delaunay ref.:** Delaunay refinement
	Coll./Split.: based on collapses and splits
Local decimation:	**Vert. rem.:** Vertex removal
	Triang. rem.: Triangle removal
Local region-merging:	**Edge coll.:** Edge collapse = iterative edge contraction,
	Vert. pair coll.: Vertex pair collapsing = vertex pair contraction
	Vert. clust.: Vertex clustering = cell collapse,
Global	

Approx.: the approximation error used

In this section we tried to give an overview of the different features necessary for the elaboration of a simplification algorithm. Most of works in the field of polygonal simplification require an operator using to simplify the geometry of the mesh, an error metric and a data structure (i.e. a hierarchical representation).These choice of these three features are the key point of simplification methods.
Operators: *While the vertex clustering operator brings some really fast robust and simple solution but with low quality results, the vertex removal and edge collapsing operators bring complex solution but with higher quality results.*
Error-based approach: *In terms of simplification process based on error measures, it appeared that the vertex-vertex distances are the fastest and most robust techniques followed by the vertex-place measures. The vertex-vertex measures provide guaranteed bounds on quality but these are typically much looser than we would like. They are also easy to implement. The vertex-plane measures produce much higher-quality models overall, but the latest quadric error measures do not guarantee a particular quality, so they do not support fidelity-based simplification systems. They are slightly harder to implement than vertex-vertex measures requiring a few more functions in your algebraic toolbox. The vertex-surface and surface-surface measures are somewhat slower than the 1st two ones simplifications but the surface-surface measure provide guaranteed error bounds on their output, making them useful for both fidelity-based and budget-based simplification systems. Surface-surface measures are difficult to implement compared to the 1st two approaches requiring more geometry-based coding rather than just a few algebraic tools.*
Hierarchical representation: *different representations are available to support progressive simplification, however this hierarchical require some storage space. While hierarchical subdivision methods are relatively fast and simple to implement and facilate the simplification process, they yield poorer quality than more general triangulation methods (for example quadtrees and quaterny triangulation can result in cracks in the surface.*
Out-of-core: *while some out-of-core algorithms have been proposed, the management of the memory stays an unresolved bottleneck.*
Appearance: *while recent appearance-preserving algorithms improved greatly the quality of the simplified mesh, these algorithms are still slow. Plus, appearance preserving simplification using vertex normals needs correctly oriented normals, which is not always the case.*

3.4 Applications

3.4.1 Data acquisition projects

3.4.1.1 Digital Michelangelo project

A team of 30 faculty, staff and students from the Stanford University and the University of Washington spent the 1998-1999 academic year in Italy scanning the sculptures and architecture of Michelangelo [Levoy2000]. The goal of this project is to produce a set of 3D computer models and make these models available to scholars worldwide. The largest of these models, the David, consists of approximately two billion polygons and requires 32GB of storage! Rendering all these polygons at once would produce many imperceptible details.

3.4.1.2 Visible Human project

The Visible Human project consists in the creation of complete, anatomically detailed, three-dimensional representations of the normal male and female human bodies. It has generated over 18,000 digitized sections of the body. This dataset of digital images are stored in MRI, CT and anatomical modes. The goal of this project is to serve as a reference for the study of human anatomy, to serve as a set of common public domain data for testing medical imaging algorithms and to serve as a test bed and model for the construction of network accessible image libraries.

3.4.2 Project on rendering and transmission

3.4.2.1 ARTE: an Adaptive Rendering and Transmission Environment

Numerous techniques have developed to permits the transmission and rendering of complex 3D models such as model compression, streaming techniques, model simplification and level of detail management. **Martin** [Martin2000] implemented a client-server system that transmits models of varying complexity to clients with different graphics capabilities, over networks, over different types of connections.

3.5 MAIN ISSUES TO RESOLVE

Different issues stay unresolved that we will describe in this section. First the size of objects continues increasing every year with the progresses of the scanner and MRI systems. Second the real-time displays of these objects are still a main issue to resolve. Then the diminution of the objects size results in a poor fidelity of the simplified model compared to the original one. Consequently it is important to find specific accurate error metric linked to this fidelity which will allow researchers to obtain a tradeoff between process/storage time and quality. To finish we will intent to list the main features to take into consideration according to the domain of application: simulation, mechanical simulation and architectural simulation.

3.5.1 Which hierarchical representation for virtual world?

The simplification of huge objects such as the David from the Digital Michelangelo project with two billion polygons has been managed by recent out-of-core researches (see section…) nevertheless RAM size is often a bottleneck and it is still needed to manage this massive size while preserving high quality simplification. Fast simplification algorithms that decrease greatly the size of the mesh, such as the vertex clustering method of Rossignac and Borrel, result often in low quality simplified models. In the opposite, some slow algorithms, such as image-driven simplifications, result in really high quality models but are computationally expensive. Consequently researchers tried to use a trade-off between the quality and the process/storage time. Thus the Quadric Error Metric of Garland et al. is one of the best balances between speed, robustness, simplicity and fidelity.

3.5.2 Which resolution for real-time visualization?

One of the main issues to resolve is to visualize in real time the object for different reasons:

- In the domain of video games: when the viewer (the player) moves around in the scene he can get further or closer from an object and thus the visualization in real-time is needed
- In the domain of CAD development: when designers construct CAD systems they need to visualize their CAD representations several times per hour and thus it is important to visualize this CAD objects on real-time in order to not loose time waiting for this display
- In the medical domain: if a surgeon uses a system that visualizes a certain part of the patient body he needs to operate then the visualization need to be done in real-time as this kind of operation need to be done quickly
- And others applications…

To manage this real-time display, different questions arise: how to manage the LOD, how to blend the transitions between each LOD switch and how to measure the accuracy of the fidelity after approximation. The real-time display depends as well on the number of polygons to visualize and by consequence on the size of the simplified mesh and it depends on the processing time necessary to obtain the simplified mesh. However according to which features we give priority, the approach will be different.

3.5.2.1 Centered approaches

The passage from a lower or higher resolution model can be determined according to different factors. In this section we classify these different factors according to the point of view we consider. In virtual worlds, we consider two different spaces: the space of the object itself and the space of the environment.

3.5.2.1.1 Object-centered

For object-centered techniques, we consider the following features:
Distance: further the object is from the viewer, fewer details are needed and thus a lower LOD can be selected that will not affect greatly the fidelity of the object. We then need a data structure of different LOD for each object and a list of distance threshold to indicate when each LOD should be used. Nevertheless as these distances are measured from the

viewpoint to an arbitrary point within the object, inaccuracy happen since the real distance to the viewpoint can change depending on orientation.
Size: further the object is from the viewer, smaller the object gets. Thus we project the object on screen and choose the LOD according to the size of this projection coverage. Thus we use a list of sizes threshold rather than distances threshold. Contrary to distance, the entire object is used so we do not need to select an arbitrary point. The result is a more generic and accurate means for modulating LOD but can also be more computationally expensive requiring a number of world coordinates to be projected into screen coordinates.
Priority: objects receive a priority rank so that the objects considered as most important are degraded the least. For example a glass on the table will be less important than the table itself. These kinds of techniques are called priority-based or semantic-based solution.
Hyteresis: in order to avoid too visible lag into the LOD transitions, the objects switch to a lower LOD slightly further away than the threshold distance and switch to a higher LOD at slightly closer distance.
Geomorphing (in geometric space)**:** in order to avoid artifacts appearing during transitions between two LODs, the geomorphing technique inserts new vertices along an existing edge and later on moves that vertex to its final position ("Morph" means that the mesh is not static but changes each frame.). It interpolates the vertices of the two LOD at every frame during the transition. Turk [Turk1992] noticed that a linear interpolation was sufficient to produce smooth transitions between models. This really complex but really efficient technique can make screen-space errors of a few pixels (which would certainly incur visible popping otherwise) appear nearly imperceptible. Hoppe [Hoppe1996] applied the term geomorph to a similar geometric interpolation between LODs in his progressive meshes algorithm.

3.5.2.1.2 Environment-centered

3.5.2.1.2.1 Light-centered

The importance of details to maintain can be defined according to the **local illumination**. To have more detail in the regions where the illumination changes sharply, detail must increase in a direction perpendicular and proportional to the illumination gradient across the surface [Xia1996]. Plus, as the back-facing polygons do not contribute to the visual realism, a **visibility culling** should be implemented to simplify these invisible regions of the object. Other **environmental conditions** can be created to show only what we want to show in the simplified scene by generating only a restricted number of LOD for each object and various environmental conditions to slacken the LOD distance threshold including clouds, fog, smoke and haze.

3.5.2.1.2.2 Eye-centered

For a better choice of the resolution of the objects in the scene, we should take into consideration the **human perception**, not the pinhole camera system. In order to choose the accurate resolution of the model, we need to ask ourselves the questions: "How much detail can we remove from the scene without the user noticing?” It is important to note that the human eyes are less sensitive to detail of an object that moves rapidly across the retina. But other factors need to be considered: the background illumination, the pupil size, the exposure time, the optical deficiencies such as myopa and age, the field of view, etc. The quantity of detail the eye can resolve is measured scientifically by the spatial frequency defined in units of cycles per degree (c/deg). While our field of view is about 200degrees horizontally and 135 degrees vertically, the human eye is particularly sensitive to detail around 8c/deg and this sensitivity drops off to zero at around 60c/deg. Our eye cannot resolve detail smaller than this limit. **Reddy** [Reddy2001] is the first researcher to have based his researches of LOD system on visual perception. He proposed a visualization system that specifies how much detail a human can perceive under various circumstances by removing detected imperceptible details in an image. Reddy processes as follow:

- He calculates highest perceptible spatial frequency at each pixel according to its velocity and distance from the fovea
- He blurs that pixel using a Gaussian filter with a kernel size equivalent to the threshold frequency

The **silhouette** plays as well an important role in the perception of detail. However, transitions between different LOD lend to distracting “popping”. To combat these artifacts researchers and developers have proposed number of smooth blending techniques with the goal of softening the transitions between levels including alpha blending (in image space) approach. The **alpha blending** assigns an opacity or alpha value with the two LODs. An alpha of 1.0 means that the object is opaque while an alpha of 0.0 means that it is invisible. This technique works as follow: Let’s imagine that we have an object with two different resolution and we want to switch from the first resolution LOD1 to the second LOD2. We define a fade range, 10m e.g., for this object. Then if the object moves away from the viewpoint and the switching distance is 100m e.g., then when LOD1 reaches 95m, both LOD1 and LOD2 are rendered at the same time until the object beyond 105m at which point only LOD2 is rendered. This technique is possible from the point of view of the eye since the LOD1 hides the second LOD. But the problem is the necessity to have two versions of the same object at the same time and thus a higher number of polygons to be rendered.

3.5.2.2 Fidelity metric

As said before, we should find an accurate metric to measure the approximation quality. Most of algorithms evaluate the error using Hausdorff distance. However this measure is either computationally prohibitively expensive either quite inaccurate. Thus we should find an error evaluation which lies on the geometrical properties of the model but the appearance as well. This error should also be defined in function of the needs of the specific domain. The simplification of the topology can be desirable for a much more aggressive geometric simplification without scarifying visual realism and can help reduce aliasing artifacts. However the topology simplification can be unacceptable in the case of mechanical CAD models and medical imaging for example. Indeed finite element analysis might require continued presence of certain topological structures such as holes and voids to determine the strength or stress at different parts of a mechanical part. And in medical imaging the data collected from computer-aided tomography (CT) or magnetic resonance imaging (MRI) scans often have important structures which should not be simplified. For this reason we need to define the domain of application before to implement this metric to know if the preservation of the topology is necessary or not.

3.5.3 Domain specific features

In this section we will try to list the main features each domain should follow. Indeed the features in simulation or in mechanical visualization or in architectural visualization will differ.

3.5.3.1 Features in simulation

The definition of important features for a simplification algorithm can differ according to the domain of application. The simulation consists in controlling the shape of the objects in the scene. Here we describe which kinds of features are studied for which kind of field.

3.5.3.1.1 Features in mechanical visualization

Memory/disk space: In any application, the memory and disk space can become a real bottleneck. It is important to find solution to store these huge amounts of data since the size of recent models can reach billions of polygons with the apparition of new scanners. In cartography the size of terrains (called height fields in computer graphics) can reach millions of points with the use of lasers. So the main goal of curve simplification in cartography (called line generalization) is to remove unnecessary detail while saving memory/disk space.

Topology preservation and geometry simplification: In mechanical visualization it is essential to preserve the topology of the original mesh. For example, in Computer Graphics, the medical and computer-aided design applications study small details of the objects. The geometry can be simplified to different LOD in the domain of video games for example. One LOD switch or blend to another appropriate LOD, making transitions as smooth as possible.

Real-time display: In computer graphics, in the domain of military flight simulators, video games or CAD for example, the real-time performances are essential. The fast display of objects or the fast redisplay after the viewpoint moves (called walkthrough) result in enhancing the realism of the scene. However for off-line processing, such as the creation of special effects, the real-time is not a main criterion while compact storage is more important.

3.5.3.1.2 Features in architectural visualization

Compact storage: Such as in the domain of cartography, saving space is vital for applications in Computer Vision where range data are acquired by stereo or structured light techniques (e.g. lasers) that produce millions of data points.

Physical fidelity: In Finite Element Analysis, the pre-process to simulation (called the mesh generation step) aims to simulate physical phenomena such as the air flow and the electromagnetic field.

Topology and geometry preservation: In architectural visualization, the topology and geometry must be preserved. For a fast simulation in Finite Element Analysis, it is important to obtain a mesh as coarse as possible while preserving physical features of interest. This kind of simplification is called mesh coarsening in this domain.

We overview in this section the different elements to take into consideration during the elaboration of a simplification algorithm. These features can differ according to the domain of application. Thus the topology-breaking can be studied in the domain of video games in order to increase the speed of the process while the topology has to be preserved in the domain of Computer-Aided Design and medical application. Some trade-off must be found between the disk/memory space requirement, the speed of the simplification and the fidelity of the result.

3.6 Conclusion

Most of simplification techniques require an operator, a data structure to sort vertices, edges, faces and an error metric to control the quality of the simplified mesh. Between the speed, robustness and simplicity of the vertex clustering algorithm [Rossignac1993] and the complexity and better looking results of the vertex removal [Schroeder1992] and edge collapsing [Hoppe1993], Garland and Heckbert [Garland1997] found a suitable trade-off between speed, robustness, simplicity and fidelity with their quadric error metrics approach. Recently, researchers took particular attention to appearance-preserving [Cohen1998] and out-of-core issues [Lindstrom2000a]. Ameliorations are still needed and this necessity increases with the increase of design progresses (scanning, etc.). The simplification features vary with the domain of application. Other factors are not been yet studied in details, but the real-time rendering of simplified objects depends as well on the visual perception, the LOD transitions smoothness, the memory storage, etc.

Among recent issues in data reduction, the semantic structures get new interest from researchers. Semantic structures such as motion parameters for virtual characters [Oh2005] can do effective role in simplification for specific purpose. In CAD domain, features like actuators should be preserved better than other parts. Those features should be identified either manually from pre-specified annotation from CAD design software or semi-automatically by identifying specific interest areas.

REFERENCES

[Algorri1996]

M. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum Eurographics'96 Proc.*, vol. 15, pp. 78–86, 1996

[Andujar1996]

C. Andujar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé. Automatic generation of multiresolution boundary representations. *Computer Graphics Forum Eurographics'96 Proc.*, vol. 15, pp. 87–96, 1996

[Bajaj1996]

C. Bajaj and D. Schikore. Error bounded reduction of triangle meshes with multivariate data. *SPIE*, 2656:34–45, 1996

[Bunny]

http://www.graphics.stanford.edu/data/

[Certain1996]

A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin and W. Stuetzle. Interactive multiresolution surface viewing. *23rd annual conference on Computer graphics and interactive techniques Proc*., 1996

[Choudhury2002]

P. Choudhury and B. Watson. Completely adaptive simplification of massive meshes. *Northwestern University tech. Report CS-02-09,* 2002

[Ciampalini1997]

A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *Technical Report C96-021, CNUCE – C.N.R. (to appear on The Visual Computer 1997),* 1996

[Cignoni1994]

P. Cignoni, L. DeFloriani, C. Montani, E. Puppo and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. *Proc. Of 1994 Symposium on Volume Visualization,* pp. 19-26, 1994

[Cignoni1998]

P. Cignoni, C. Rocchini and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17,* vol. 2, pp. 167-174, 1998

[Cignoni2003]

P. Cignoni, C. Montani, C. Rocchini and R. Scopigno. External memory management and simplification of huge meshes. *IEEE Transactions on Visualization and Computer Graphics 9(4),* pp. 525-537, 2003

[Clark1976]

J. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM,* vol. 19(10), pp. 547-554, 1976

[Cohen1996]

J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. *ACM Computer Graphics SIGGRAPH'96 Proc.,* pp. 119–128, 1996

[Cohen1998]

J. Cohen, M. Olano and D. Manocha. Appearance-preserving simplification. *ACM Computer Graphics SIGGRAPH'98 Proc.,* 1998

[Cohen-Or1999]

D. Cohen-Or, D. Levin, and O. Remez. Progressive compression of arbitrary triangular meshes. In *IEEE Visualization'99 Proc.*, pages 67–72, 1999

[DeFloriani1984]

L. De Floriani, B. Falcidieno, G. Nagy, and C. Pienovi. A hierarchical structure for surface approximation. *Computer Graphics* 8, pp.183-193, 1984

[DeFloriani1989]

L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Application,* vol. 9, No. 2, pp. 67-78, 1989

[DeFloriani2004]

L. De Floriani, P. Magillo, E. Puppo and D, Sobrero. A multi-resolution topological representation for non-manifold meshes. *Computer Aided Design Journal,* 36(2), pp. 141-159, 2004

[Douglas1973]

D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *TheCanadian Cartographer*, 10(2):112–122, 1973

[Eck1995]

M. Eck, T. De Rose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *ACM Computer Graphics SIGGRAPH'95Proc.,* pp. 173–181, 1995

[El-Sana2000]

J. El-Sana, Y. Chiang. External memory view-dependent simplification. *Computer Graphics Forum 19(3),* pp. 139-150, 2000

[Erikson1999]

C. Erikson and D. Manocha. GAPS: General and automatic polygonal simplification. *Proceedings of 1999 ACM Symposium on Interactive 3D Graphics,* pp. 79-88, 1999

[Fandisk]

http://research.microsoft.com/research/graphics/hoppe/

[Femur]

http://miles.cnuce.cnr.it/cg/homepage.html

[Garland1997]

M Garland and P.S. Heckbert. Surface simplification using quadric error metrics. *ACM Computer Graphics SIGGRAPH'97 Proc.*, 1997

[Garland1998]

M. Garland and P. Heckbert. Simplifying Surfaces with Color and Texture using Quadric Error Metrics. *IEEE Visualization 98 Proc.,* 1998

[Garland2001]

M. Garland, A. Willmott and P. Heckbert. Hierarchical face clustering on polygonal surfaces. *ACM Symposium on Interactive 3D Graphics.* 2001

[Garland2002]

M. Garland and E. Shaffer. A multiphase approach to efficient surface simplification. *Visualization'02,* pp. 117-124, 2002

[Gomez1979]

D. Gomez and A. Guzman. Digital model for three-dimensional surface representation. *Geo-Processing,* 1:53-70, 1979

[Gotsman2003]

C. Gotsman, X. Gu and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *SIGGRAPH 2003 Proc.,* 2003

[Gross1996]

M. Gross, O. Staadt and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Trans Visual. and comp. Graph.,* 2(2):130-144, 1996

[Gu2002]

X. Gu, S. Gortler and H. Hoppe. Geometry images. *SIGGRAPH 2002,* pp. 355-361, 2002

[Gu2003]

X. Gu and S. Yau. Global conformal surface parameterization. 2003

[Gueziec1996]

A. Guéziec. Surface simplification inside a tolerance volume. *Technical Report RC 20440, I.B.M. T.J. Watson Research Center*, 1996

[Hamann1994a]

B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197-214, 1994

[Hamann1994b]

B. Hamann and J. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design 11,* pp. 477-489, 1994

[He1995]

T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. *IEEE Visualization'95 Proc.*, pp. 296–303, 1995

[He1996]

T. He, L. Hong, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Trans. on Visualization & Computer Graphics*, 2(2):171–183, 1996

[Hinkler1993]

P. Hinkler and C. Hansen. Geometric optimization. *IEEE Visualization'93 Proc.,* pp.189-195, 1993

[Hoppe1993]

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *ACM Computer Graphics SIGGRAPH'93 Proc.,* pp. 19–26, 1993

[Hoppe1996]

H. Hoppe. Progressive meshes. *ACM Computer Graphics SIGGRAPH'96 Proc.,* pp. 99–108, 1996

[Hoppe1997]

H. Hoppe. View-dependent refinement of progressive meshes *ACM Computer Graphics SIGGRAPH'97 Proc.*, pp. 189-198, 1997

[Hoppe1998]

H. Hoppe. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. *IEEE Visualization'98*, pp. 35–42, 1998

[Hoppe1999]

H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. *Proceedings in IEEE Visualization'99,* pp. 59-66, 1999

[Isenburg2003]

M. Isenburg, P. Lindstrom, S. Gumhold and J. Snoeyink. Large mesh simplification using processing sequences. *Visualization'03,* pp. 465-472, 2003.

[Kalvin1996]

A. Kalvin and R. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE C.G.&A*, 16(3):64-77, 1996

[Khodakovsky2003]

A. Khodakovsky, N. Litke and P. Schroder. Globally smooth parameterization with low distortion. *SIGGRAPH 2003,* 2003

[Kim1998]

H. Kim, S. Jung and K. Wohn. A multiresolution control method using view directional feature. *VRST 1998,* 1998

[Kim2001]

H. Kim and K. Wohn. Multiresolution model generation with geometry and texture. *VSMM 2001*, 2001

[Kim2004]

H. Kim and K. Wohn. AGSphere: multiresolution structure of directional relationship on surface parts. *IEICE Trans. Inf. & Syst.,* vol. E87-D, 2004

[Klein1996]

R. Klein, G. Liebich and W. Straβer. Mesh reduction with error control. *7th IEEE Visualization'96 Conference,* pp. 311-318, 1996

[Kobbelt1998]

L. Kobbelt, S. Campagna, J. Vorsatz and H. Seidel. Interactive multi-resolution modeling on arbitrary meshes. *SIGGRAPH'98 Proc.,* pp. 105-114, 1998

[Kumler1994]

M. Kumler. An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). *Cartographica*, 31(2), Summer 1994. Monograph 45, 1994

[Levoy2000]

M. Levoy et al., "The Digital Michelangelo Project: 3D Scanning of Large Statues," *Computer Graphics* (Siggraph 2000 Proc.), ACM Press, New York, Aug. 2000, pp. 131-144.

[Levy2002]

B. Lévy, S. Petitjean, N. Ray and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21,* vol. 3, pp. 362-371, 2002

[Lindstrom1998]

P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *IEEE Visualization 98 Conf. Proc.,* pp. 279-286, 544, 1998

[Lindstrom2000a]

P. Lindstrom. Out-of-core simplification of large polygonal models. *SIGGRAPH 2000,* pp. 259-262, 2000

[Lindstrom2000b]

P. Lindstrom and G. Turk. Image-driven simplification. *ACM Transactions on Graphics,* vol. 19(3), pp. 204-241, 2000

[Lindstrom2001]

P. Lindstrom and C. Silva. A memory insensitive technique for large model simplification. *Visualization'01,* pp. 121-126

[Lounsbery1994]

J. Lounsbery. Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *PhD thesis*, Available as ftp://cs.washington.edu/pub/graphics/LounsPhd.ps.Z, 1994

[Low1997]

K Low and T. Tan. Model simplification using vertex-clustering. *ACM SIGGRAPH'97, http://www.iscs.nus.sg/~tants/,* 1997

[Luebke1997]

D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. *ACM Computer Graphics SIGGRAPH'97 Proc*., vol. 31, pp. 199-208, 1999

[Maillot1993]

J. Maillot, H. Yahia and A. Verroust. Interactive texture mapping. *SIGGRAPH 1993*, pp. 27-34, 1993

[Martin2000]

I. Martin. ARTE – An adaptive rendering and transmission environment for 3D graphics. *8th ACM International Conference on Multimedia*, pp. 413-415, 2000

[MetroTool]

http://vcg.isti.cnr.it/downloads/downloads.htm

[Nooruddin1999]

F. Nooruddin and G. Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics,* vol. 9, No. 2, 1999

[Oh2005]

Seungwoo Oh, HyungSeok Kim, Pascal Volino, Nadia Magnenat-Thalmann and Kwangyun Wohn, Semantics over Geometry: Garmented Body Model Generation for Real-time Simulation, Workshop towards Semantic Virtual Environments (SVE 2005), Villars, Switzerland, March, 2005

[Popovic1997]

J. Popovic and H. Hoppe. Progressive simplicial complexes. *ACM Computer Graphics SIGGRAPH'97 Proc.,* pp. 217-224, 1997

[Praun2003]

H. Praun and H. Hoppe. Spherical parameterization and remeshing. *SIGGRAPH 2003 Proc,* 2003

[Prince2000]

C. Prince. Progressive meshes for large models of arbitrary topology. *Master's thesis,* 2000

[Reddy2001]

M. Reddy. Perceptually optimized 3D graphics. *IEEE Computer Graphics and Applications,* vol. 21(5), pp. 68-75, 2001 www.martinreddy.net/percept

[Renze1996]

K. Renze and J. Oliver. Generalized unstructured decimation. *IEEE C.G.&A.*, 16(6):24–32, 1996

[Ronfard1996]

R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum Eurographics'96 Proc.*, 15(3):67–76, 1996

[Rossignac1993]

J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. *Geometric Modeling in Computer Graphics*, pp. 455–465, 1993

[Samet1984]

H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys,* vol. 16, Issue 2, 1984

[Sander2001]

P. Sander, J. Snyder, S. Gortler, H. Hoppe. Texture mapping progressive meshes. *ACM Computer Graphics SIGGRAPH 2001 Proc.*, pp. 409-416, 2001

[Schroder1992]

W. Schroder, J. Zarge and W. Lorensen. Decimation of triangle meshes, *ACM Computer Graphics SIGGRAPH'92 Proc.*, vol. 26, pp. 65-70, 1992

[Shaffer2001]

E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. *IEEE Visualization 2001 Proc.,* 2001

[Soucy1996]

M. Soucy and D. Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. Computer *Computer Vision and Image Understanding*, 63(1):1–14, 1996

[Taubin1998]

G. Taubin and J. Rossignak. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2), 1998

[Touma1998]

C. Touma and C. Gotsman. Triangle mesh compression. *Graphics Interface '98 Proc.*, pp. 26–34, 1998

[Turk1992]

G. Turk. Re-tiling polygonal surfaces. *ACM Computer Graphics SIGGRAPH'92 Proc.*, vol. 26, pp. 55–64, 1992

[VonHerzen1987]

B. Von Herzen and A. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics SIGGRAPH'97 Proc.,* 21(4): 103-110, 1987

[Wu2003]

J. Wu and L. Kobbelt. A stream algorithm for the decimation of massive meshes. *Graphics interface'03,* pp. 185-192, 2003

[Xia1996]

J. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. IEEE Visualization'96, pp. 327-114, 1996

Bibliography for important surveys

[Alliez2003]
P. Alliez and C. Gotsman. Recent advances in compression of 3D meshes. *Symposium on Multiresolution in Geometric Modeling,* 2003

[Cignoni2004]
P. Cignoni, L. De Floriani, P. Lindstrom, J. Rossignac and C. Silva. Multi-resolution modeling, visualization and streaming of volume meshes. Eurographics Tutorial 2, 2004

[Cignoni1998]
P. Cignoni, C. Montani and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics,* 22(1):37-54, 1998

[DeFloriani2004]
L. De Floriani. A survey on data structures for LOD models. To appear

[Franc2002]
Martin Franc. Methods for polygonal mesh simplification. State of the art and concept of doctoral thesis. 2002

[Garland1999]
M. Garland. Multiresolution modeling: survey and future opportunities. *Eurographics'99,* 1999

[Heckbert1997]
S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. *Multiresolution surface modeling courses notes, ACM SIGGRAPH'97,* 1997

[Luebke2002]
D. Luebke, M. Reddy, J. Cohne, A. Varshney, B. Watson and R. Huebner. Level of detail for 3D graphics. Morgan-Kaufmann Publishers, 2002. http://lodbook.com/

[Luebke2001]
D. Luebke. A developer's survey of polygonal simplification algorithms. Tutorial, 2001

[Puppo1997]
E. Puppo. Simplification, LOD and multiresolution principles and applications. *EUROGRAPHICS'97,* vol. 16, N. 3, 1997

[Schroder2002]
P. Schroder. Subdivision as a fundamental building block of digital geometry processing algorithms. 2002

Available Code

Links to available code are grouped together on this webpage: http://lodbook.com/source/
We also add these two links:

www.martinreddy.net/percept by Reddy 2001 to visualize the limits of vision
http://www-users.informatik.rwth-aachen.de/~roberts/software.html

Glossary

Chart:

See patch

Decimation methods:

Work bottom-up. Opposite of the refinements methods, they start with a triangulation of all of the input points and iteratively delete elements (typically vertices) from the triangulation gradually simplifying the approximation. They start with a polygonisation (typically triangulation) and successively simplify it until the desired level of approximation is achieved. The advantage is that it can be generalized to volumes.

Distortion metric:

If the input triangle has three edges a1, a2, and a3 and the output triangle has three edges b1, b2, and b3 then we calculate Li = max{ai,bi} and li = min{ai,bi} and the distortion is equal to d = max{L1/l1; L2/l2; L3/l3}. The distortion is always superior or equal to 1. A distortion equal to 1 is equivalent to say that the triangles are isometric.

Genus:

The genus of the surface X is defined as X = V – E + F (Euler's formula) where V is the number of vertices of a polygonal mesh, E is the number of edges and F is the number of faces.

Harmonic map:

It approximates the conformal map for fixed boundaries

Hyteresis:

A lagging or retardation of the effect, when the forces acting upon a body are changed, as if from velocity or internal friction; a temporary resistance to change from a condition previously induced, observed in magnetism, thermoelectricity, etc., on reversal of polarity.

Incremental:

Operation on a set of data that are the data only added since the last manipulation. Starts with the original model M0, iterative simplification algorithms generate a sequence of approximations: M0->M1->…->Mk arriving at the final approximation. Attention!!! Differs from the progressive mesh notation used by Hoppe where M0 is the base mesh which, by a sequence of refinements is transformed into the original mesh Mk. An incremental representation is one which encodes the original model M0, the final model Mk and all the intermediate approximations M1,…, Mk-1.

Non-regular:

Non-regular models are models of mixed dimensionality.

Parameterization:

Parameterization is the process of mapping a surface onto regions of the plane. The specification of a curve, surface, etc., by means of one or more variables which are allowed to take on values in a given specified range

A shape can be parameterized using different tools:

- Discrete approach (fictious load)
- Bezier & B-Spline curves
- Uniform B-Spline (NURBS)
- Feature-based solid modeling (in CAD)

Patch / chart:

Region homeomorphic to a disc

Per-vertex:

A per-vertex color means that the color is assigned to a vertex only and not to the entire face. The color of the face is obtained by an interpolation between vertices. And it is the same for per-vertex normals.

Refinement methods:

Work top-down. Refinement methods are multi pass algorithms that begin with a minimal initial approximation (a coarse approximation), and that insert on each pass one or more elements as vertices in the triangulation, and repeat until the desired error is achieved or the desired number vertices is used.

Simplicial complex: mathworld

Simplicial complex is a space with a triangulation. Formally, a simplicial complex K in $\mathbb{R}^n$ is a collection of simplices in $\mathbb{R}^n$ such that

1. Every face of a simplex of K is in K, and
2. The intersection of any two simplices of K is a face of each of them

View-independent:

The simplification is not dependent on the spectator's position and viewing angle and need not to be recalculated if the spectator moves. The spectator can not be included into the simplification process causing that invisible or far parts of the scene can not be considered differently than near parts.

View-dependent:

The simplification considers the spectator's position and viewing angle and has to be partially or fully recalculated if the spectator moves. The advantage of view-dependent algorithms is that invisible or far parts of the scene can be simplified more than near parts. This sometimes increases the visual quality of the result incredibly.