An approach of quality simplification of 3D model using MAYA API

Prof. Yogesh Singh[1] Prof. B.V.R.Reddy[2] Mr. R. Rama Kishore[3]
[1]Professor, USIT, Guru Gobind Singh Indraprastha University, Delhi 06;
[2] Professor, USIT, Guru Gobind Singh Indraprastha University, Delhi 06;
[3]Assistant. Professor, USIT, Guru Gobind Singh Indraprastha University, Delhi 06;

Abstract
Computer Graphics applications are passing through the problem of having complex polygonal surface models for the limited available hardware capacity. These models cause large processing time and less transmission speed. There is a trade off between quality of these models and processing time. As we improve the quality of mesh the processing time increases and vice-versa. It is proposed to address this problem by simplifying the surface while maintaining its quality at run time. The algorithm uses iterative contractions of edges to simplify models. The main issue of edge contraction is to decide which edge to be contracted. This is answered by the use of quadric error matrices. Each edge is assigned with a quality attribute based on quadric error technique and the edges will be deleted on the "least quality contract first" basis. The proposed concept is implemented in MAYA API programming and complex models can be easily simplified through plug in.
Keywords:
surface simplification, edge contraction, quadric error metric, MAYA API.

1. Introduction

Polygonal surface models are used by computer graphics applications for display and simulation purposes. These models have large set of polygons; and to get more realism more numbers of polygons are required. But large amount of polygons i.e. large detail will challenge the hardware capability and lead to increase the processing time and transmission time. A focus is required to address this problem and to propose to simplify the high detailed polygon model into much simplified model. Simplified model should contain fewer numbers of polygons without compromising the quality of original model.
In medical, scientific, entertainment, and computer aided design systems; polygonal models are commonly used for representation. In all these applications and many more, highly detailed and complex models are generally required. But to achieve acceptable running times and hardware capacity, simpler model should substitute the original detailed model. Recent work on polygonal simplification algorithms has focused on this goal. Different methods have been proposed in this context, like vertex clustering, polyhedral refinement, region merging, wavelet decomposition, decimation of vertex and contraction of edges. Care should be taken to find the cost of each edge and the least cost edge should be contracted first.
Some predominant geometry like boundary features should not be simplified in order to sustain the basic quality of model. An error measuring techniques may be used to estimate the quality of the model at every step of simplifying algorithm.

2. Related work

2.1 Error metric methods:

Error metric methods are used for assigning a quality attribute to every edge of the polygonal model.
Some of the error metric methods are:

2.1.1 Hausdroff error metric:

Hausdroff error was used in order to improve the quality of polygonal simplification by Klein et al. [1] Hausdroff distance is a distance metric between point sets. Given two sets of points, A and B, the Hausdroff distance is defined as

$H(A, B) = \max(h(A, B), h(B, A))$
Where $h(A, B) \neq h(B, A)$.

Because a surface is a particular type of continuous point set, Hausdroff distance provides a useful measure of the distance between two surfaces.

2.1.2 Quadric error metric:

Quadric error metric was used by Garland and Heckbert [2] for extending the quality of simplification. While calculating the error instead of taking only points on the surface for consideration, this uses geometric local information of the input surface.
Each face in the original model defines a plane, which satisfies the equation
$nT. v+d=0$

Manuscript received February 5, 2010
Manuscript revised February 20, 2010

where n=[nx,ny,nz]T is a unit normal vector and d is a constant. The squared distance of a vertex v=[x,y,z] to this plane is given by

D2 = [nT.v+d]2= (v Tn+d) (nTv+d) = vT(n nT)v+2dnTv+d2.

This is a quadratic form, plus linear term, plus a constant. We can conveniently represent D2 using a quadric Q

Q = (a,b,c)

Therefore, if we are merging two vertices v1 and v2, the resulting quadric is merely the sum

Q = Q1+Q2

This defines the cost of a contraction error. This estimates about the distance error between the simplified and original object.

2.1.3 Discrete differential error metric:

Discrete differential error metric is based on the theory of local differential geometry in such a way that the first and second order discrete differential approximation is done locally. It is proposed by Sun–Jeong Kim [3]. Though previous error metric methods give visually pleasing results with a reasonably fast speed, it is hard to measure an accurate geometric error on highly curved and thin regions since error measured by distance metric is small in such cases and causes a loss of visually important features. To overcome such drawback, first and the second order discrete differentials are approximated to take care of slope and curvature preservation of surface features.

2.2 Surface simplification methods:

Surface simplification methods are broadly classified into two categories:

- Refinement: These types of algorithms begin with coarse approximations and refine the approximation by adding up elements at each step.
- Decimation: It is a reverse of refinement. The algorithms that use this methodology start with the fully detailed surface (finest level) and extracts elements at each step iteratively.

Some of the polygonal simplification methods are:

2.2.1 Manual Preparation

Different details of surface model are constructed through human hand. Hence the process becomes more time consuming and having a lot of overheads.

2.2.2 Polyhedral Refinement

These algorithms use the refinement methodologies in background. But the decimation methodology has been much more widely used, because there are some practical difficulties associated with the refinement process. The coarse approximation must have the same topology as the original one which cannot be easily discovered in the beginning.

2.2.3 Vertex Clustering

The Vertex clustering algorithm was proposed by Rossignac and Borrel [4]. The cluster in which all the vertices have existed is divided uniformly. All vertices within each cluster are unified to a single representative vertex. Numerous vertex clustering algorithms have been proposed. Low and Tan [5] have proposed modified algorithm that uses floating clustering in place of uniform clustering, which enhances the consistency of simplification. Kanaya, Teshima, Nishio and Kobori [6] have proposed topology preserving simplification algorithm that uses depth first search tree on a vertex clustering algorithm. But it was observed that using clustering methods, the degree of simplification obtained is not satisfactory.

2.2.4 Region Merging

Simplification algorithms [7, 8] are based upon region merging. Kalvin & Taylor [7] have proposed superfaces algorithm. It uses bounded approximation approach which describes that a simplified model approximates the original one to a pre-specified tolerance. Hinker & Hanson [8] have proposed Geometric optimization. They gave an application independent algorithm that merges coplanar and nearly coplanar polygons into large polygons and then re triangulated into simpler polygons.

2.2.5 Wavelet Decomposition

According to Stollnitz, DeRose, Salesin [9], wavelet decomposition functions can be used for decomposing a surface into series of details. That's why; this method is an efficient way to produce multi-resolution modeling. Kin, Valette, Jung & Prost [10] have proposed local wavelet decomposition for 3D surfaces. Their approach is an extension of the work of Lounsbery et al. He has proposed the wavelet decomposition only for regular triangular mesh subdivision.

2.2.6 Vertex Decimation

It was firstly proposed by Schroeder et al [11]. He proposed an algorithm that reduces the number of triangles by removing vertices which are selected in the decimation process. All the triangles associated with that vertex are also removed and the total shape is re-triangulated (fig. 1). Algorithm of Schroeder has some deficiency in preserving smooth surfaces. These surfaces will become quite rough during process of simplification. Renze and Oliver [12] have proposed generalized surface decimation that uses vertices as a primitive element for removal. Franc and Skala [13] proposed a mesh decimation algorithm without sorting. Through this, we can achieve a fast algorithm for mesh simplification in parallel environment.

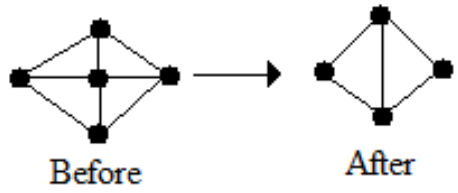


Figure 1: Vertex Decimation.

2.2.7 Iterative Contraction of edges

Hoppe et al [14] and co-workers have used edge collapse with swapping of edges and their splitting for simplification of surfaces. The algorithm that will describe in this paper is also based on this method.
When an edge is contracted, its end points are replaced by a single point and all the triangles that are associated with that edge are removed and the resulting model is re-triangulated (figure 2).

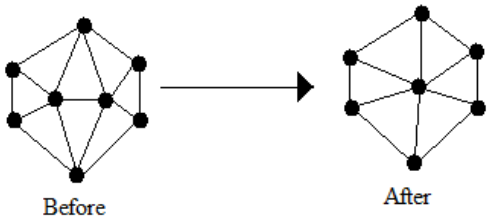


Figure 2: Edge Contraction.

Hoppe et al [14] used a distance measure for determining target vertex positions. Ronfard and Rossignac [15] have proposed an algorithm for approximation of a polyhedral object at different detail levels. The algorithm collapses the edges on the basis of deviation from initial shape. The algorithm was based upon local incremental operations but they also kept track of the history of the original surface. Garland and Heckbert [2] have developed a surface simplification algorithm that contract vertex pairs by making use of quadric error metric technique. According to Jia-xin CHEN and Hai-he HU [16] the standard quadric error metric proposed by Garland can lead to inaccurate simplification, they proposed their algorithm that also uses edge contraction method and improved quadric error metric as background.

2.3 Implementations through MAYA API

2.3.1 Manipulation of Elastically Deformable Surfaces through Maya Plug-in:

Efforts were made to develop a mathematical model from the theory of plate bending in elasticity which relates physical properties of a surface to its elastic deformation. It presents the finite difference solution of the mathematical model and implements it using the Maya API [17] and the MEL [18] scripting language. It was examined the effects of material properties and other factors on surface shapes and demonstrate applications of the proposed approach in controlling the shape of elastically deformable surfaces.

2.3.2 Virtual Scene for Telerobotic Operation:

Efforts were made on building virtual modeling environments which can provide instant visualization for Humanoid Teleoperation system as the visual feedback is an essential part of any telerobotic system. Here virtual scene includes reconstruction of Humanoid robot BHR02, and objects like table etc, where as reconstruction of BHR02 and interface for rendering the data of real robot was already developed [19]. The main goal of this work is to enhance our visual teleoperation system for BHR02 developing virtual environment includes objects like table, etc to avoid any collision during real time operation. 3D modeling software Maya is used for modeling and simulations. Maya plug-ins in VC++ provides efficient modeling convention, real time interaction, and time saving rendering approach in a virtual environment. In this paper we describe the overview of method of creating geometrical shape of objects and rendering the data of scene in visual environment.

3. MAYA API

3.1 Introduction

MAYA API is used to provide the MAYA software with new capabilities and functionalities. It uses a set of C++ classes to embed these functionalities in MAYA.[20]

3.2 Overview of MAYA architecture

MAYA is open source software, which is very flexible and extensible in nature. At lower level architecture, MAYA comprises of a database called dependency graph (DG) for storing graphical information in *objects* called *nodes*. These nodes are characterized by different types of attributes. Data flows from one node to another, by connecting similar type of attributes. In spite of the flexibility and working power of DG, it limits the operations on a scene. To overcome this limitation, MAYA provides approximately 900 commands. These commands performs various functions like creation of DG nodes, setting and connecting their attributes, and creating transform nodes, which define the positioning of elements on the scene. Some of these commands are used for creation of user interfaces like building windows, menus, buttons etc. Also, if the user needs to add a non-existing MAYA command, it can be easily accomplished with the help of MAYA API, which is used as a plug-in.

3.3 Overview of DAG hierarchy

In Maya, elements such as position, orientation and scale of geometry are defined by a directed acyclic graph (DAG). This DAG comprises of two type of nodes viz. transform nodes, and shape nodes.

3.3.1 Transform Node:

It maintains the parenting as well as the transformation (translation, rotation, scaling etc.) information.

3.3.2 Shape Node:

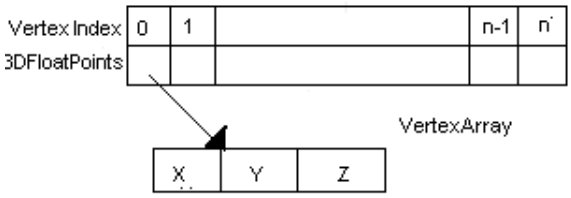
It maintains all the geometrical information. A shape node does not maintain transformation related information, so it needs a transform node directly above it to support a particular geometry. Thus all geometry requires two nodes, a shape node directly above it, and a transform node above the shape node.

3.4 The Polygon API

To handle polygon geometry in Maya, Maya API provides a Polygon API as its subset. It uses some basic data structures for representation of polygon components like faces, edges, vertices etc. Encapsulation of these data structures into polygon shape nodes forms the dependency graphs, which is the core of Maya architecture.
Polygon components: Polygon meshes are comprised of three basic components viz. faces, vertices and edges, and two additional components viz. face-vertices and UVs.

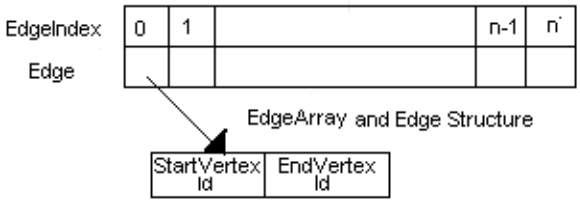
3.4.1 Vertices:

A vertex array data structure stores the vertices of a polygon mesh in the form of 3D float points corresponding to each entry of a vertex-id. All the edges and faces correspond to this array.



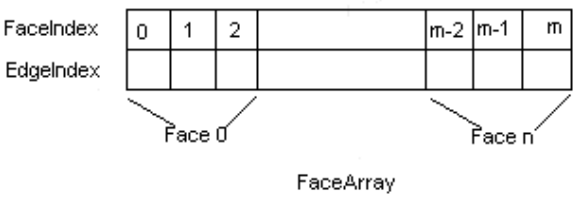
3.4.2 Edges:

An edge array data structure is used to store edges of a polygon mesh in a two integer format. Corresponding to each entry of an edge-id in the edge array, there resides a (start vertex-id, end vertex-id) integer pair which defines the starting and ending vertices of an edge. It also provides vertex composition and direction.



3.4.3 Faces:

An integer array is used to store the faces of a polygon mesh. The indices of the array represent the edge id's and each face is made up of a number of sequences of integers corresponding to these indices. The boundary of the face is represented by first sequence of edges. Holes in the face are represented by any subsequent sequences. Finally, the internal flags are used for marking the start and end of each sequence, and the end of a face description.



3.4.4 Face-Vertices:

When two or more faces are adjacent to each other, they share common vertices. Such vertices are known as face-vertices. They came into existence for fulfilling the need of associating data to a vertex of a specific face, while distinguishing the same vertex from any faces that it share. A face array data structure is used to store face vertices. These are conceptual features used by the polygon features such as color per vertex and UVs.

4. Proposed Simplification Algorithm

4.1 Approximation through quadric error metric

To contract an edge there must be some criteria upon which edge is chosen to be collapsed; i.e. there is a requirement of assigning cost to every edge. To define cost, quadric error metric is used.

4.2 Algorithm overview

Simplification method consists of repeatedly selecting the edge with a minimum cost, collapsing this edge, and then re-evaluating the cost of edges affected by this edge collapse. The cost of the edge is evaluated using quadric error metric technique.

The algorithm itself can be summarized as follows:

1. Calculate quadric error metric for each vertex (Jia-xin CHEN and Hai-he HU [16]) using

$$Q_{v_i} = \left(\sum_{i=1}^{n} f_i Q_i \right) \Big/ n$$

Where
n = number of triangular planes associated with the vertex Vi.
Qi = Standard quadric error metric
fi = Area of triangular plane of order i.

2. Extract all edges from source model.

3. Assign a cost of contraction to each edge.

Cost of edge: $\Delta\overline{V} = \overline{V^T}\,\overline{Q_e}\,\overline{V}$
Where

$\overline{V}$ = Target vertex made after collapse.

Q_e = Quadric error metric for edge

$$Q_e = Q_{vi} + Q_{vj}$$

Q_{vi}, Q_{vj} = Quadric error metric for vertices connected to the edge e.

4. Put the edges in priority queue and sort them based on the cost of contraction.

5. Repeat until desired approximation is reached.

- Remove the edge(i, j) with the least cost from the queue.
- Contract this edge in to the single vertex $\overline{V}$, update the mesh neighborhood.

$\overline{V}$ = Position which minimizes the cost of edge $\Delta\overline{V}$. Thus $\overline{V}$ is calculated by solving $\frac{\partial\Delta}{\partial x} = 0$, $\frac{\partial\Delta}{\partial y} = 0$, $\frac{\partial\Delta}{\partial z} = 0$.

- Update costs for all edges connected to Vi, Vj.

The standard quadric metric can be used to calculate both the cost of a contraction and target position of the vertex. According to Jia-xin CHEN and Hai-he HU [16] the standard quadric error metric, as we know that in 3D model there are some triangles associated with every vertex. Because of the process of simplification it may be possible that number of triangles to a vertex become so much so that the error of that vertex will be naturally higher. This phenomenon leads the cost of peripheral vertices to be lower than middle vertices. Thus peripheral vertices will be chosen to simplify and lead to inaccurate simplification and inaccurate visual effect.
This is caused because the standard quadric error is defined by summation. It was proposed to make the quadric error in form of average error and add the area of triangular areas. Hence, the cost of internal vertices will not be larger than outer vertices. Now the quadric error metric of each vertex is expressed as:

$$Q_{v_i} = \left(\sum_{i=1}^{n} f_i Q_i \right) \Big/ n$$

where n = number of triangular planes associated with the vertex Vi.

Qi = Standard quadric error metric
fi = Area of triangular plane of order i.

4.3 Derivations for quadrics

The average quadric error can be determined by using the formula:

$$Q_{v_i} = \left(\sum_{i=1}^{n} f_i Q_i \right) \Big/ n \qquad 1.$$

To get this quadrics we must have the value of fi and Qi.

4.3.1 Calculation for Qi which is standard quadric error metric:

Plane P = $[a, b, c, d]^T$

Where a, b and c are the x, y and z component of the normal and d value of the plane equation represents the distance of the plane to the origin only when the normal is unit length.

Vertex V = $[x, y, z, 1]^T$

The error of the vertex ΔV = sum of squared distances to its planes (the planes of the triangles that meet that vertex). It is the cost of that vertex.

Distance from vertex to Plane

$$= \frac{ax + by + cz + d}{\sqrt{a^2 + b^2 + c^2}}$$

Here $a^2 + b^2 + c^2 = 1$

So Distance = $ax + by + cz + d$

Distance^2 = $(ax + by + cz + d)^2$

$$\text{Distance}^2 = \left(\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} x & y & z & 1 \end{bmatrix} \right)^2$$

Distance^2= $(P^T V)^2$

$$\text{So } \Delta V = \sum_{P \in \text{Planes at V}} (P^T V)^2$$

$$\Delta V = V^T Q_i V$$

Here K_P = Quadric error metric for plane P.

And Q_i = standard quadric error metric for vertex $V_{i.}$ which is sum of all K_P.

As $K_P = PP^T$

$$K_P = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} a & b & c & d \end{bmatrix}$$

$$Q_i = \sum_{p \in \text{Planes at V}} \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

Let Q_i be $\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$ ……2.

4.3.2 Calculation for fi :

As f_i is the area of plane related to vertex i. and we know the plane is triangular so according to the Heron's formula the area of a triangle is:

$$f_i = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{……3.}$$

Where a, b, c are the sides of the triangle

And $s = \frac{a + b + c}{2}$

4.3.3 Final Quadric error metric for vertex Vi :

The quadric error metric for vertex $V_{i.}$ from equation 1=

$$Q_{v_i} = \left(\sum_{i=1}^{n} f_i Q_i \right) \Big/ n$$

by substituting the values of f_i and Q_i from equation 2 and 3 into equation 1 we get =

$$Q_{V_i} = \left(\sum_{i=1}^{n} \left(\sqrt{s(s-a)(s-b)(s-c)} \right) * \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \right) / n \qquad \text{….4.}$$

4.3.4 Cost of edge:

The cost of edge = $\Delta\overline{V} = \overline{V^T}\,\overline{Q_e}\,\overline{V}$ …..5.

Where

$\overline{V}$ = target vertex made after collapse.

$\overline{Q_e}$ = Quadric error metric for edge

$\overline{Q_e} = Q_{vi} + Q_{vj}$ …….6.

Q_{vi}, Q_{vj} = Quadric error metric for vertices connected to the edge e.

4.3.5 Determining the position of target vertex $\overline{V}$ made after collapsing the edge:

After contracting the edge a target vertex is made let's suppose it is $\overline{V}$. Now the issue is to determine the position of vertex $\overline{V}$. $\overline{V}$ will be the position of the vertex which minimizes the cost of edge which is $\Delta\overline{V}$. From equation 5 we have

$$\Delta\overline{V} = \overline{V^T}\,\overline{Q_e}\,\overline{V}.$$

Now maxima and minima approach is used to find out the location of vertex $\overline{V}$.

$$\overline{V} = [x, y, z, 1]^T$$

Minimization of $\Delta\overline{V}$ is:

$$\Delta\overline{V} \Rightarrow \frac{\partial\Delta}{\partial x} = 0, \frac{\partial\Delta}{\partial y} = 0, \frac{\partial\Delta}{\partial z} = 0$$

From equation 6 we have

$$\overline{Q_e} = Q_{vi} + Q_{vj}$$

Let suppose the value of $\overline{Q_e}$ is:

$$\overline{Q_e} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix}$$

as $\Delta\overline{V} = \overline{V^T}\,\overline{Q_e}\,\overline{V}$.

$$\Delta\overline{V} = [x, y, z, 1] \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\Delta\overline{V} = \begin{aligned} & q_{11}x^2 + q_{12}xy + q_{13}xz + q_{14}x + q_{21}yx + q_{22}y^2 + \\ & q_{23}yz + q_{24}y + q_{31}zx + q_{32}zy + q_{33}z^2 + q_{34}z + \\ & q_{41}x + q_{42}y + q_{43}z + q_{44} \end{aligned}$$

……..7.

Now minimization process:

Partial differentiation of equation 7:

(i) $\frac{\partial\Delta}{\partial x} = 0$ (ii) $\frac{\partial\Delta}{\partial y} = 0$ (iii) $\frac{\partial\Delta}{\partial z} = 0$

By equation 8, 9 and 10 we get

$$\begin{bmatrix} q1 & q2 & q3 & q4 \\ q5 & q6 & q7 & q8 \\ q9 & q10 & q11 & q12 \\ q13 & q14 & q15 & q16 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} q1 & q2 & q3 & q4 \\ q5 & q6 & q7 & q8 \\ q9 & q10 & q11 & q12 \\ q13 & q14 & q15 & q16 \end{bmatrix} \overline{V} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Hence position of target vertex will be =

$$\Delta\overline{V} = \begin{bmatrix} q1 & q2 & q3 & q4 \\ q5 & q6 & q7 & q8 \\ q9 & q10 & q11 & q12 \\ q13 & q14 & q15 & q16 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

5. Conclusion

A surface simplification algorithm which is capable of producing quality approximations of complex polygonal models developed through MAYA API plug-in is proposed. Iterative edge contractions are used for simplification. Quadric error metric is used to determine the error of simplification. Further the paper describes the following:

Quadric Error Metric: Quadric error metric is used for assigning a quality attribute to every edge of the polygonal model to sustain the overall shape of the model.
Surface Simplification Algorithm: The edge collapse method is used as a surface simplification algorithm. By combining the quadric error metric with edge collapse we'll be benefited with a fast and quality based approximation. This also requires less hardware capacity.
MAYA API: The implementation of algorithm can be done using MAYA API programming. Polygonal models can be easily simplified through plug-in. This implementation will be very beneficial as it reduces processing time, hardware requirement and in fast transmission of 3 D models while working on heavy model of network.

6. Future Scope

The proposed algorithm can efficiently be implemented. The implementation of the surface simplification can be beneficial for the software applications for 3D digital animation and visual effects like MAYA, 3D Max etc. One can create like this feature as plug-in and effectively work on 3D models with less processing time and capacity requirements.

References

[1] Klein, Reinhard, Gunther Liebich, and Wolfgang Straßer, "Mesh Reduction with Error Control", Proceedings of IEEE Visualization '96.

[2] Garland, Michael and Paul Heckbert, "Surface Simplification using Quadric Error Metrics", Proceedings of SIGGRAPH 97. pp. 209-216.

[3] Sun-Jeong Kim, Soo-Kyun Kim and Chang-hum Kim, "Discrete Differential Error Metric for Surface Simplification", computer Graphics and Applications Proceedings, 276-283, 2002.

[4] Jarek Rossignac, Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. In Geometric Modeling in Computer Graphics , pp. 455-465, Springer Verlag, Eds. B. Falcidieno and T.L. Kunii, Genova, Italy, June 28-July 2, 1993.http://www.gvu.gatech.edu/~jarek/Papers/VertexClustring.pdf

[5] Kok-Lim Low, Tiow-Seng Tan. Model Simplification Using Vertex-Clustering. In 1997 http://portal.acm.org/citation.cfm?id=253310&dl=acm&coll=portal

[6] Takayuki Kanaya, Yuji Teshima, Koji Nishio, Ken-ichi Kobori. A Topology-Preserving Polygonal Simpli?cation Using Vertex Clustering. In 2005 http://portal.acm.org/ft_gateway.cfm?id=1101410&type=pdf

[7] Alan D. Kalvin, Russell H. Taylor. Polygonal Mesh Simplification with Bounded Error. In 1996 http://ieeexplore.ieee.org/iel1/38/10541/00491187.pdf http://portal.acm.org/citation.cfm?id=618962

[8] Paul Hinker, Charles Hanson. Geometric optimization. In 1993 http://ieeexplore.ieee.org/iel5/2945/28178/01260766.pdf?tp=&arnumber=1260766&isnumber=28178

[9] Eric J. Stollnitz, Tony D. DeRose, David H. Salesin. Wavelets for Computer Graphics. In 1995 http://portal.acm.org/citation.cfm?id=616037.618291&coll=portal&dl=ACM&CFID=15151515&CFTOKEN=6184618

[10] Yun-Sang Kim, Sebastien Valette, Ho-You Jug, and Remy Prost. Local Wavelets Decomposition for 3-D Surfaces. In 1999 http://ieeexplore.ieee.org/iel5/6632/17687/00819612.pdf?arnumber=819612

[11] William J. Schroeder, Jonathan A. Zarge, William E. Lorensen. Decimation of triangle meshes. In 1992 http://portal.acm.org/citation.cfm?id=134010

[12] Kevin J. Renze, James H. Oliver. Generalized Surface and Volume Decimation for Unstructured Tessellated Domains. In 1996 http://ieeexplore.ieee.org/iel2/3517/10587/00490518.pdf?arnumber=490518

[13] Martin Franc, Vaclav Skala. Parallel Triangular Mesh Decimation Without Sorting. In 2001 http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=945333

[14] Hugues Hoppe, Tony DeRose, Tom Duchamp,John McDonald, Werner Stuetzle. Mesh Optimization. http://research.microsoft.com/~hoppe/meshopt.pdf

[15] R. Ronfard and, J. Rossignac. Full-range approximation of triangulated polyhedra. In 1996, http://perception.inrialpes.fr/publications/1996/rrr96/fullrange01.pdf

[16] Jia-xin CHEN, Hai-he HU. One Mesh Model Simplification Method Based on Shape Transform of Triangles. IEEE Computer Society, 2006. http://ieeexplore.ieee.org/iel5/4089190/4089191/04089302.pdf

[17] L.H. You,Javier Romero Rodriguez,Jian J.Ahang.,"Manupulation of Elastically Deformable Surfaces through Maya Plug-in", Proceedings of the Geometric Modelling and imaging- New Trends,IEEE2006.

[18] A.M.Day,D.B.Arnold, S.Havemann, D.W. Fellner, "Combining Polygonal and subdivision Surface approaches to modeling and rendering of urban environments",Computers &Graphics 28(2004) 497-507. ELSEVIER

[19] M. Usman Keerio, A.Fattah Chandio, A. Khawaja and A Raza Jafri , " Virtual Scene for Telerobotic Operation", International Conference on emerging Trends IEEE2006.

[20] Maya 8.0 manuals, Discreet product.