International Journal of Automation, Control and Intelligent Systems
Vol. 1, No. 1, 2015, pp. 1-8
http://www.publicscienceframework.org/journal/ijacis

Polygonization Method for Functionally Defined Objects

Sergey I. Vyatkin*

Synthesizing Visualization Systems Laboratory at Institute of Automation and Electrometry, Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russian Federation

Abstract

Obtaining a triangular mesh from a set of cells intersecting the surface with perturbation functions is considered in this paper. The seamless rendering is discussed. The free forms based on the analytical perturbation functions are presented. A method for polygonization freeform surfaces is proposed.

Keywords

Seamless Rendering, Polygonization, Perturbation Functions, Octree, Polygonizer

Received: April 6, 2015 / Accepted: April 13, 2015 / Published online: April 20, 2015

1. Introduction

Several representations of geometric objects are currently used in computer graphics. Each of the objects, according to its properties, is used in different fields, beginning from 3-D simulation and CAD systems up to real-time visualization systems.

The functional representation describes most accurately the object geometry and has the smallest size of the required data. Procedures of functional representation demonstrate compact and flexible representation of surfaces and objects that are the results of logical operations on volumes. Its disadvantage is complicated geometrical processing and visualization in real-time.

There are seven kernels: Gaussian function [1, 2 and 3], Inverse function, Inverse squared function and Soft objects [4, 5], Metaballs [6], W-shaped quartic polynomial and Cauchy function [7, 8].

Tessellation or polygonization is the generation of polygonal approximation of a function-based surface. Why need tessellation functionally defined objects? Conventional graphics environment are optimized for polygon display and manipulation. Many geometric calculations (such as finite element analysis) are performed on polygonal models. Navigable nature of polygons (list of vertices, edges, faces) supports non-graphical operations such as placing an object on a surface. Systems of rapid prototyping operate with polygonal models.

However, there are difficulties in achieving this task. The generation of polygons from function-based form (for example, implicit form) is less straightforward than from parametric form. Quadric surfaces can be converted to the parametric form; it is difficult to do so for higher order functions.

In general, polygonization software or a polygonizer must evaluate the function in sample points and employ some numerical procedure to locate polygon vertices. There are several algorithms known as tessellation, polygonization, triangulation, tiling and meshing [9, 10].

This paper a method of tessellation of functionally defined

* Corresponding author
E-mail address: sivser@mail.ru

objects with any desired precision is proposed. Octree local object space subdivision for search of points of a surface was used. As a result program calculates all points of a surface without holes [11].

Organization

The paper is organized as follows. Section 2 gives a brief survey of the work related to polygonization of implicit surfaces. Section 3.1 describes free forms based on perturbation functions. Section 3.2 provides the overview of approach to computation of the voxel-surface intersection. The octree algorithm that performs efficient retrieval of volume elements (voxels) involved in surface points finding. Linking surface points into polygons is described in Section 3.3. Hard edges described in this paper are given in Section 3.4. The export of resulting mesh and examples of results are considered in Section 4. Summary and future work is discussed in Section 5.

2. Previous Works

2.1. Grids and Cells

A polygonizer uses a spatial partitioning a 3D volume divided into semi disjoint (adjacent but non-overlapping) cells that completely enclose the function-based surface, in particular, implicit surface [12]. Each cell intersected by the implicit surface (a transverse cell) is polygonized: intersections of the cell edges with the surface are connected to form a set of polygons. A fixed resolution (or uniform space) means that the cells are fixed in size. An adaptive resolution means that the cell size is locally proportional to the size of surface detail. The accuracy of fixed resolution methods depends on the size of the cell: large cells remove detail and small cells yield a lot of polygons. Large cell size is used to produce low-resolution polygons in real-time for rendering on graphics processors with near real-time design interaction. Small cell size is used for quality visualization with direct rendering methods such as ray tracing.

2.2. Classes of Polygonizers

There are two classes of polygonizers, depending on the input data. Discrete data available at cell corners (voxel data sets). Space partitioned by exhaustive enumeration [13]. Continuous data is the function can be evaluated at arbitrary point in space. Space partitioned by subdivision or by numerical continuation. Discrete data can be represented in the continuous form with the trilinear (or other) interpolation. For discrete data the function behavior between the cell corners must be approximated (linear interpolation in the edge). For continuous data, a function value in any point can be determined with given accuracy.

2.3. Spatial Partitioning Schemes

A spatial partitioning divides space into semi-disjoint cells that collectively enclose the surface; the interior of the object is not enclosed. The cells may then polygonized. Subdivision is the recursive division of space into subvolumes that fully enclosed the surface. There are the following types of trees: octree [14, 15 and 16], bintree [17], KD-tree [18].

A set of cubes of different size can fill space without gaps or overlaps. A cube may be subdivided into similarly shaped and oriented polyhedra (eight subcubes). A specially shaped tetrahedron, known as the Kuhn complex, may be subdivided into similar subshapes: eight subsimplices, four at the corners and four internal to the original one.

A class of incremental polygonization techniques usually divided into predictor-corrector and piecewise-linear methods [19, 20 and 21]. Predictor-corrector methods apply directly to the surface creating polygons by joining an initial surface point with additional points [22]. Generation of new points: predicted position - displacement along the tangent plane; corrected position with Newton iteration. Works well for contours but problematic for surfaces (contour points are ordered). Piecewise-linear methods begin with a single transverse seed cell. Enclose the surface incrementally by semi disjoint cells. A seed cell center (seed point) can be located by a random search within a slowly increasing radius of some starting point. New cells are the neighbors sharing the transverse faces. The process iterates until the entire surface is enclosed. To prevent cycling, the processed cell locations must be stored. An implicit surface can consist of several disjoint components. Continuation produces only a single surface component for one seed cell. To polygonize all components, appropriate seed cells have to be selected (automatically or by a user). Simplicial continuation employs not cubes but n-dimensional simplices and consists of a single simplex and rules to compute adjacent simplices. Finally, the implicit surface is surrounded by tetrahedra (volumetric triangulation), for example, by Kuhn simplices. A polygonization method for non-manifold is surfaces was introduced in [23].

2.4. Cell Polygonization

Cell polygonization generates a set of polygons for the surface patch inside a single transversal cell. This procedure consists of the following steps. Detected a cell edge which intersects the surface (different function signs in the endpoints). Such edge is assumed to contain a single intersection. Compute an intersection point (surface vertex). Connect surface vertices to form polygons. Surface vertex computation consists from linear interpolation and binary subdivision. Surface vertices connection consists of the

following steps. Algorithm starts with a transversal edge, looks for the next transversal edge in the face and stops when the polygon is complete. Table for tetrahedra is as follows. From a four-bit number abcd such that bit a is set to 1, if the function is positive in vertex a. The sixteen cases are obtained. Table for cubic cells ("Marching Cubes") is as follows [13]. The configuration of the set of polygons for a cubic cell depends on the number of cell corners with positive function values. For eight corners, there are two hundred fifty six possible configurations. Only fifteen configurations have to be stored. Others are equivalent to them due to symmetry and rotations.

2.5. Ambiguities

Ambiguity occurs for certain configurations at the cell level. Alternate surface vertex connection for a cell face. There are ambiguous corner configurations for a cube. Surface vertices must be connected consistently along shared faces. Holes appear in the method [13]. Disambiguation strategies are topology inference, preferred polarity, and cell decomposition. Topology inference determines vertex connectivity using different schemes: face center sampling [4], approximated function gradients [24]. Preferred polarity ensures that the polygon separates cell corners of a certain sign [25]. Cell decomposition subdivides a cell into tetrahedra to resolve the ambiguity [25].

Particle-based techniques were proposed in [26].

3. Method Description

3.1. Basics

A functionally defined object is completely defined by means of the real-valued describing function of three variables (x_1, x_2, x_3) in the form of $F(X) \geq 0$, then the objects are treated as closed subsets of the Euclidean space E^3, defined by the describing function $F(X) \geq 0$, where F is the continuous real-valued function and $X= (x_1, x_2, x_3)$ is the point in E^3, defined by the coordinate variables. Here $F(X) > 0$ defines points inside the object, $F(X) = 0$ defines points on the boundary, and $F(X) < 0$ defines points that lie outside and do not belong to the object.

This is the basis for tessellation.

It is possible to describe complex geometry forms by specifying surface deviation function (of second order) in addition to surface basic function of second order [27]. Generally a function F(x,y,z) specifies surface of second order that is quadric (see Fig. 1):

$$F(x,y,z) = A_{11}x^2 + A_{22}y^2 + A_{33}z^2 + A_{12}xy + A_{13}xz + A_{23}yz + A_{14}x + A_{24}y + A_{34}z + A_{44} \geq 0, \quad (1)$$

where x, y and z are spatial variables.

Figure 1. Surface of second order

The free form is a composition of the base surface and the perturbation functions

$$F'(x,y,z) = F(x,y,z) + \sum_{i=1}^{N} f_i R_i(x,y,z) \quad (2)$$

where fi is the form-factor; the perturbation function $R(x, y, z)$ is found as follows

$$R_i(x,y,z) = \begin{cases} Q_i^3(x,y,z), if\ Q_i(x,y,z) \geq 0 \\ 0, if\ Q_i(x,y,z) < 0 \end{cases} \quad (3)$$

Herein, $Q(x, y, z)$ is the perturbing quadric.

Since $\max[Q + R] \leq \max[Q] + \max[R]$, for estimating the maximum Q on some interval we have to calculate the maximum perturbation function on the same interval. The obtained surfaces are smooth (see Fig. 2), and creation of complex surface forms requires few perturbation functions.

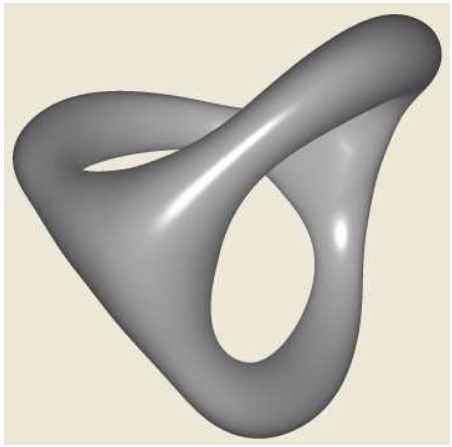

Figure 2. The free form based on quadric (synthesized by means of the three analytical perturbation functions)

3.2. Surface Points Building

When intended for triangulation no perspective normalization is used thus the scene is located in local cube space. Points are building by recursive subdivision of local space. At every step program subdivides local space into eight parts and stop when achieved maximum level of octree. For every part of volume can say whether this part belongs to object or is outside of object. If part of space is outside of the object program stops subdivision and skip this part. Thus program continue to subdivide space where object is located. Just like after subdividing stage when algorithm ends we have a buffer, some elements of which empty while others contain object subdivision results.

Octal subdivision coefficients of function-based surfaces are divided into four. And the coefficients XYZ are divided into two. This reduces the size of the cube (4).

$$\begin{aligned} A' &= A/4, \\ B' &= B/4, \\ C' &= C/4, \\ D' &= D/4, \\ E' &= E/4, \\ F' &= F/4, \\ G' &= G/2 + i*A/2 + j*D/4 + k*E/4, \\ H' &= H/2 + i*D/4 + j*B/2 + k*F/4, \\ I' &= I/2 + i*E/4 + j*F/4 + k*C/2, \\ K' &= K + i*G/4 + j*H/4 + k*I/4, \end{aligned} \tag{4}$$

Then made the shift to vector (+/- 0.5, +/- 0.5, +/- 0.5), i.e. in one of the eight subcubes. The necessary resolution defined by the user. The calculation of points is carried out for one angle surface this is sufficient for the lack of artifacts, since no holes are formed in the resulting search locations octree subdivision of local space object. And, hence, no need to adjust the local curvature.

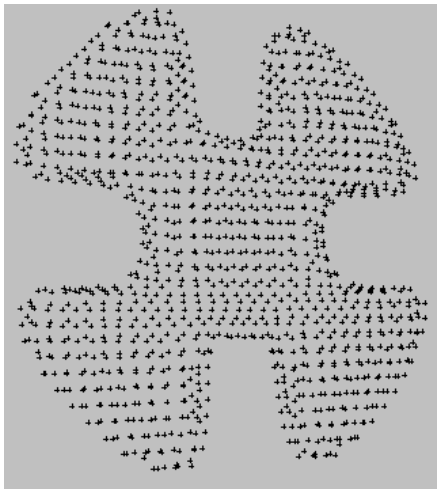

Figure 3. The cloud of points

As result we got cloud of points, which consists of surface (Fig. 3).

3.3. Linking of Surface Points into Four-Sided Polygons

Every surface point belongs to the vertical and horizontal cross-section. The current task is to find point's neighbors in cross-section. In figure 4 neighbors of surface point B are surface points A and C.

We search neighbors in horizontal and vertical cross-sections.

Figure 4. Vertical cross-section of ellipsoid. Black points are interior points, red points are surface

Algorithm of finding neighbors for point in vertical cross-section (for horizontal cross-section algorithm is analogous) is presented below.

For selected point P we build initial vector of movement from empty voxel near P on the ray to surface point P (dY, dZ) (See Fig. 5).

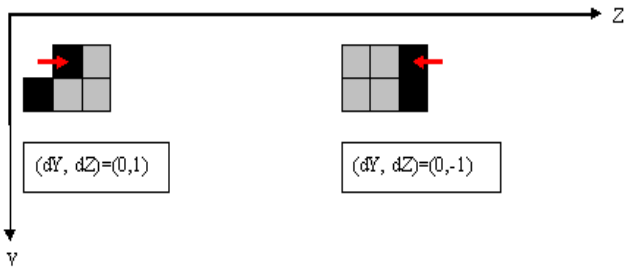

Figure 5. Initial vectors for surface points; surface points are black points; interior points are gray points

At each step we look up the cells around in the following order: right, front, left, back relatively to vector (See Fig. 6).

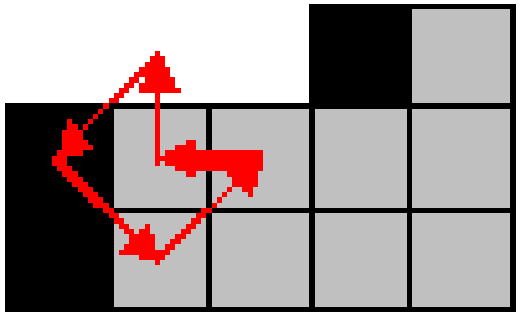

Figure 6. Looking up for cells

We select the first cell which is not empty namely Q. Then we reconstruct vector dY=Q.Y-P.Y; dZ=Q.Z-P.Z.

Now we move to the Q point and have current vector equals (dY, dZ).

So we move to the border of cross-section.

If Q point is surface point – we stop, because the neighbor is found.

If Q is interior cell – we continue looking up the cells (Q)

If we look up cells in the *right, front, left, back* order then we move by the border contour of vertical cross-section counter-clockwise. If we look cells in opposite order: left, front, right, back, we move by the border clockwise.

So for each point P we could find Up-Neighbor (moving clockwise) and Down-Neighbor (moving counter-clockwise).

For horizontal cross-section we have cross-sections in XZ plane and vector has coordinates (dX, dZ). Initial enter vector we find the same way as described above (dX=0, dZ= -1 or 1). If we move by horizontal cross-section border clockwise we find Left-Neighbor, if counter-clockwise – we find Right-Neighbor.

For linking surface points into 4-sided polygons we move from surface point A down in A's vertical cross-section to A's Down-Neighbor B; then from B in B's horizontal cross-section to it's Right-Neighbor C; then from C in C's vertical cross-section to it's Up-Neighbor D; then from D in D's horizontal cross-section to it's Left-Neighbor P; if P is A we enclose 4-sided polygon ABCD.

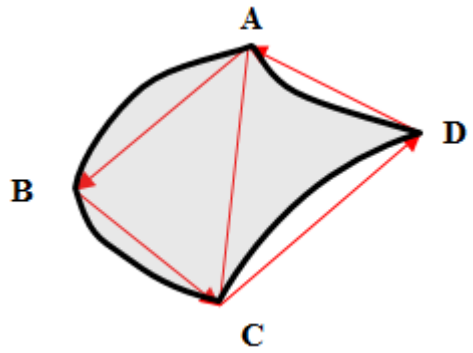

Figure 7. Linking of surface points A, B, C, D into 4-sided polygon ABCD

We do this for every surface point.

We produce two faces by A->B->C and C->D->B. Their normal orientation can be either inward or outward; so please note that currently you need to see meshes as *double-sided* object to see all faces independently their orientation.

The time of linking is O(N), where N is amount of builder surface points.

The result is a first grid quadrangles, and after the division of quadrangles - a grid of triangles (See Fig. 8).

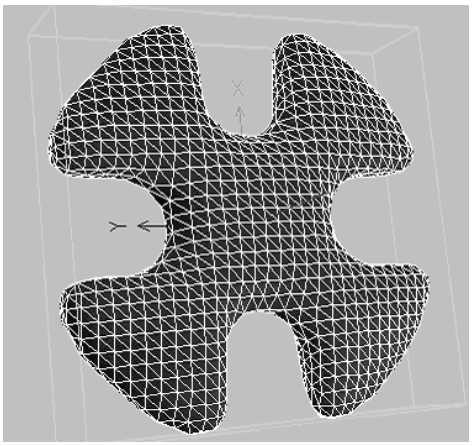

Figure 8. A grid of triangles

3.4. "Hard" Edges

On "hard edges", which belong two faces with wide angle (over 45 degree) between normals, relaxation of the edge after the insertion in the face of centroid (mass center, crossing the medians) can bring about the loss of the form (see Figure 10- is the mesh of cone.). So for "hard edge" PQ instead of relaxation of the edge is done partition:

1) Criterion of partition: if s(P) is a desired average length of the edges in the point P, s(Q) is a desired average length of the edges in the point Q, that values of lengths of the edges, which we want to reach, is conducted test:

If $\sqrt{}(2.0)||PQ|| > [s(P)+s(Q)]/2$ that is the edge too long, is conducted partition

2) If the edge must be split on criterion in 1, the point S, splitting the edge has coordinates $S=(s(P)Q+s(Q)P)/(s(P)+s(Q))$

That is to say S splits an edge on the center only then, when s(P)=s(Q). If s(P) is not s(Q), the point S splits an edge PQ pro rata weights.

3) After partitioning of the edge each face is divided on two faces (see Figure 9).

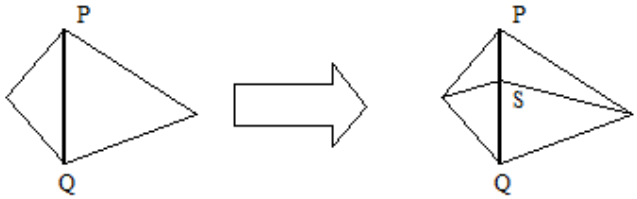

Figure 9. There is tight of holes on sides of cone

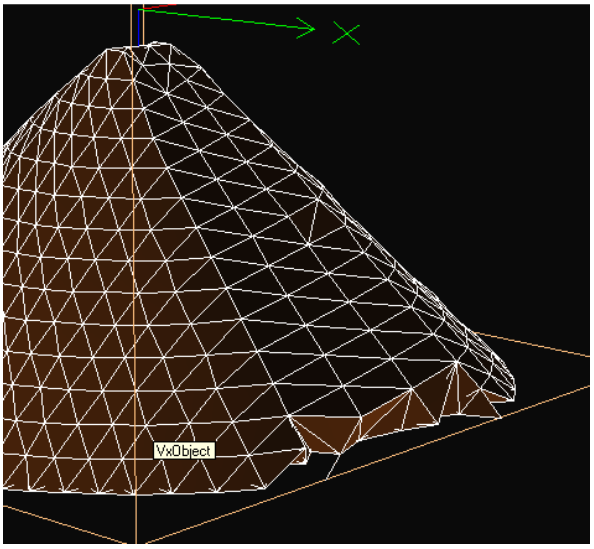

Figure 10. The triangle mesh of the cone with artifacts

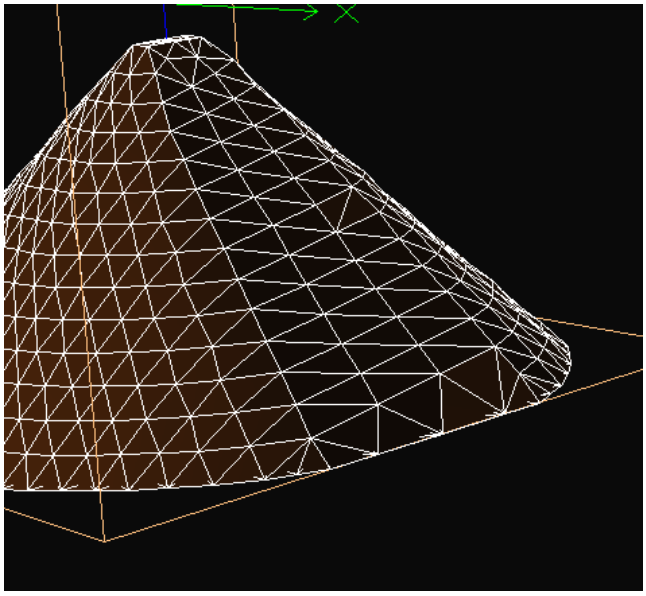

Figure 11. The triangle mesh of the cone without artifacts

4. Export of Resulting Mesh

Export mesh in ASE format was used. ASE format is text format (ASCII) and there are many converters from ASE format to other format 3DS, OBJ, X etc. In order to see exported model correctly please check double-sided property for the model, because some faces can be oriented inward. You can export object in ASE format from *VXDemo* by standard *File->SaveAs* dialog (Fig. 12).

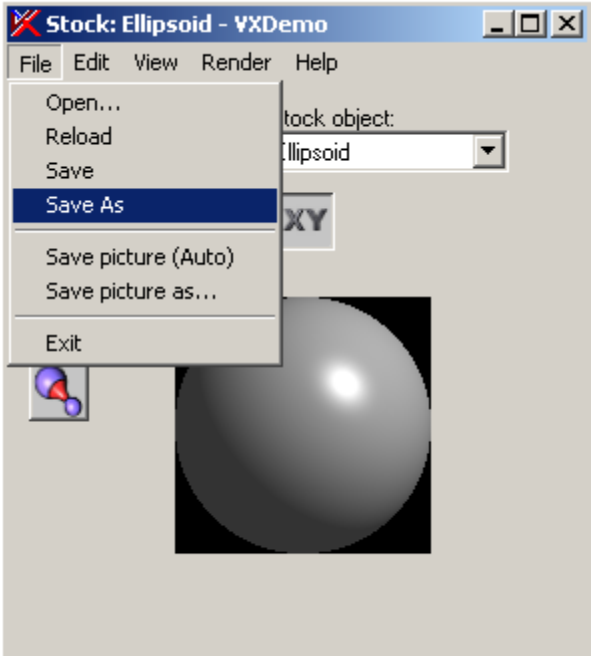

Figure 12. VxDemo export

Select .ase from the drop-down list of available extensions and enter name of file to export to (Fig. 13).

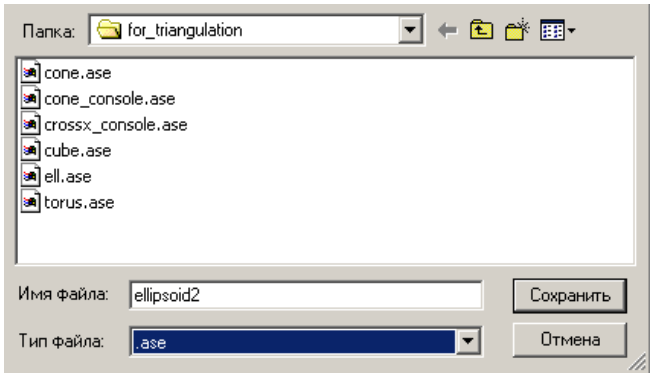

Figure 13. Export object in ASE format

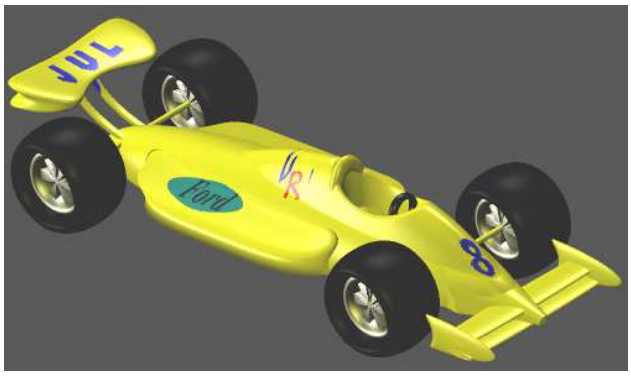

Figure 14. Function-Based object: Perturbation Functions Amount=83

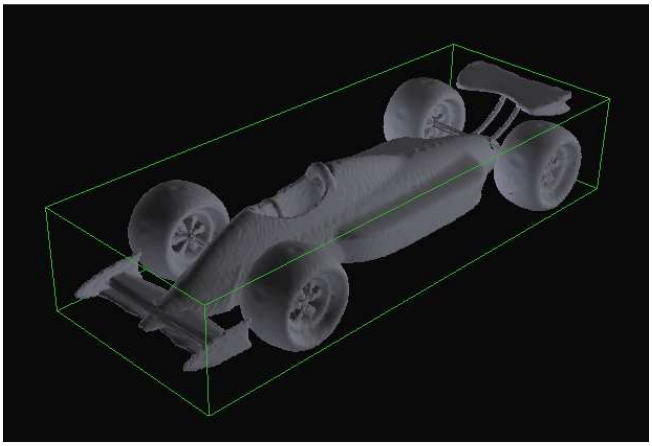

Figure 15. Polygonal object: Faces Amount=136306 Vertices Amount=68418

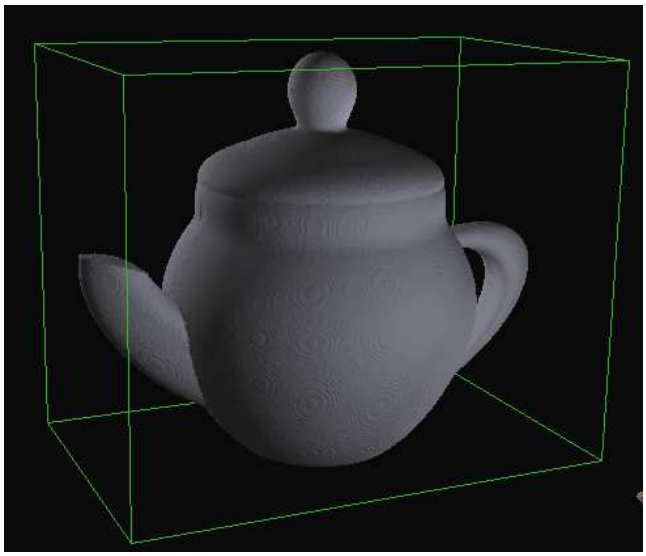

Figure 16. Model of teapot: Faces Amount=317188 Vertices Amount=158613 Perturbation Functions Amount=13

Figures 14, 15 and 16 show the result of tessellation.

For analyzing how close the function-based surface the polygonal mesh were used two error metrics. The metrics measure deviations of the vertices from the function-based surface and deviations of mesh normal from the normal of the function-based surface.

Estimation of two error metrics and uniformity is considered: tessellation of perturbation functions directly - uniformity; criteria – vertex deviation, normal deviation, uniformity.

5. Summary

Octree subdivision mesh extraction method based on recursion is proposed. A cell containing the model is subdivided into eight smaller cells, which are examined whether they contain the surface or not. The process repeats, until the minimum cell size is reached. All the leaf cells containing the surface are polygonized. Unlike the predictor-corrector methods, this algorithm does not need an initial point to begin the polygonization with. It detects the surface automatically without holes.

Proposed algorithm of surface points building works faster than step-by-step algorithm. Step-by-step calculations look every cell of space to define whether object is inside or outside the cell, but proposed algorithm can skip big amount of empty cells at once. During the surface triangulation, a mesh is calculated from a multitude of points. This mesh may have holes. The main stages of the many known method are: hole identification, hole triangulation, mesh refinement, and mesh fairing.

There are several areas for future work. The performance of algorithms can be further improved by investigating more optimizations.

References

[1] J. F. Blinn. A generation of algebraic surface drawing. ACM Transactions on Graphics, 1(3): 235-256, July 1982

[2] Bloomenthal J., Shoemake K., “Convolution surfaces”, SIGGRAPH’91, Computer Graphics, vol.25, No.4, 1991, pp. 251-256.

[3] Bloomenthal J. Modeling the mighty maple. Computer Graphics, 19(3): 305-311, July 1985.

[4] Wyvill G., McPheeters C., Wyvill B. Data structure for soft objects. The Visual Computer, Vol. 2, No. 4, 1986, pp. 227-234.

[5] G. Sealy, G. Wyvill. Smoothing of three dimensional models by convolution. In Computer Graphics International’96, June 1996, pp. 184-190.

[6] Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. The Transactions of the Institute of Electronics and Communication Engineers of Japan, J68-D(4): 718-725, 1985.

[7] A. Sherstuyk. Fast ray tracing of implicit surfaces. In Implicit Surfaces’98, pp. 145-153, June 1998.

[8] McCormack J., Sherstyuk A. Creating and rendering convolution surfaces, Computing Graphics Forum, vol. 17, No.2, 1998, pp. 113-120.

[9] Ning P., Bloomental J. An evaluation of implicit surface tilers. IEEE Computer Graphics and Applications, Vol. 13, No. 6, 1993, pp. 33-41.

[10] Bloomental J. et al. Introduction to implicit surfaces (Eds.), Morgan Kaufmann Publishers, 1997, 332 pp.

[11] P. Liepa. “Filling Holes in Meshes”// Proc. Eurographics/ACM SIGGRAPH Symp. Geometry Processing, 2003, pp. 200-205.

[12] J. Bloomenthal. An Implicit Surface Polygonizer. Graphics gems IV, Academic Press Professional, Inc. San Diego, CA, USA 1994, pp. 324-349.

[13] Lorensen W., Cline H. Marching Cubes: a high resolution 3D surfaces construction algorithm. SIGGRAPH'87, Computer graphics, Vol. 21, No. 4, 1987, pp. 163-169.

[14] G.M. Hunter and K. Steiglitz. Operations on Images Using Quad Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2):145–153, 1979.

[15] C.L. Jackins and S.L Tanimoto. Oct-trees and their use in representing three dimensional objects. Computer Graphics and Image Processing, 14(3):249–270, 1980.

[16] I. Gargantini and H.H. Atkinson. Ray Tracing an Octree: Numerical Evaluation of the First Interaction. Computer Graphics Forum, 12(4):199–210, 1993.

[17] J. L. Bentley. Multidimensional binary search trees used for associative search. Communications of ACM, 18(9):509-516, 1975.

[18] A. W. Moore. An introductory tutorial on kd-trees Extract from Andrew Moores PhD Thesis: Efficient Memory-based Learning for Robot Control. PhD Thesis; Technical Report No 209. Computer Laboratory. University of Cambridge. 1991.

[19] T.Karkanis, A.J.Stewart, Curvature dependent triangulation of implicit surfaces, IEEE Computer Graphics Application 2, 2001, pp. 60-69.

[20] E. Hartmann, A marching method for the triangulation of surfaces, The Visual Computer, Springer, 14:3, 1998, pp. 95-108.

[21] E. Galin, S. Akkouche, Incremental polygonization of implicit surfaces, Graphic Models and Image Processing, 62, 2000, pp. 19-39.

[22] S. Akkouche, E. Galin, Adaptive implicit surface polygonization using Marching Triangles, Computer Graphics Forum. 20:2. 2001. pp. 67-80.

[23] J. Bloomenthal and K. Ferguson. Polygonization of Non-Manifold implicit surfaces. In Robert Cook, editor, SIGGRAPH '95 Conference Proceedings, Annual Conference Series, pp. 309-316. ACM SIGGRAPH, Addison Wesley, August 1995.

[24] J. Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. ACM SIGGRAPH Computer Graphics, Volume 24 Issue 5, Nov. 1990, pp. 57-62.

[25] Bloomenthal J. Polygonization of implicit surfaces. Computer-Aided Geometric Design, Vol. 5, No. 4, 1988, pp. 341-355.

[26] A. Witkin, P. S. Heckbert. Using particles to sample and control implicit surfaces, SIGGRAPH 94, 1994, pp. 269-277.

[27] S.I. Vyatkin. Complex Surface Modeling Using Perturbation Functions// Optoelectronics, Instrumentation and Data Processing. – 2007 –Vol. 43, N3. pp. 226-233.