Progressive Refinement with Topological Simplification: A Theoretical Framework

ERGUN AKLEMAN [*]
Visualization Laboratory
College of Architecture

JIANER CHEN
Department of Computer Science
College of Engineering

Abstract

This paper presents a theoretical framework for progressive refinement of manifold meshes with topological simplification. We demonstrate that topology changes are not intuitive and therefore a great deal of care is necessary for handling topological simplification. We illustrate non-intutive nature of topology changes with several examples. We also show how to use the non-intutive nature of the topology changes as an advantage and develop a theretical framework for progressive refinement with topological simplification.

1 Introduction

Our goal in this paper is to develop a theoretical framework for progressive refinement schemes with topological simplifications. Level-of-detail (LOD) representations have recently become popular in computer graphics [9, 14, 10, 19, 18, 17]. Among the LOD representations progressive refinement is particularly useful since it allows continuous change [18, 17]. Although topological simplification can provide better simplification and efficient representation, there exists no framework that provides both progressive refinement and topological simplification. Ignoring topological simplification is not special to progressive meshes. In fact, except one recent development (Hybrid meshes) [12] topological simplification is not even considered in other LOD representations. Moreover, Hybrid meshes is a user-assisted LOD approach and cannot automatically guarantee topological simplification.

As an example of application of progressive refinement with topology simplification, let us consider a city that consists of huge number of buildings or a forest that is made up of lots of trees. It is not really helpful to simplify each building or each tree to a simple polyhedron. At the end, there will still be a huge number of polyhedra. The most viable alternative is to combine several buildings to create one composite building and to combine a group of trees to create one composite tree. In this way, from the distance a city block that consists of a set of buildings will be represented by one composite building and a cluster of trees will be represented by one composite tree. Moreover, this refinement should be progressive, i.e. the change must be gradual. For instance, let us say that the forest has 100 trees. The number of trees should gradually reduce; $99, 98, 97, \ldots, 5, 6, 4$ and so on. At the simplest level the whole city or forest should be represented by a single building or tree. Then this building or tree should further be progressively refined into simplest polyhedra that resemble the shape of the building or tree.

This example demonstrate progressive refinement with topology simplification will definitely be beneficial for a wide variety of applications that includes games, mesh data transmission, rendering.

In this paper, we develop a theoretical framework for progressive refinement with topological simplification. We will demonstrate that topology changes are not intuitive and therefore a great deal of care is necessary for handling topological simplification. We show that the same operation can both close a hole and cut a handle. Moreover, all topology change operators can both complicate and simplify the topology. Our framework uses the non-intutive nature of topology changes as its advantage and guarantees high compression rates with topological simplication.

1.1 Previous Work

Rendering complex geometric models at interactive rates has always been a challenging problem in computer graphics. Improving the rendering performance alone could not solve this problem since the complexity of geometric models increases with rendering performance. In other words, as the rendering performance improves, more and more complicated geometric models will be used. It is observed

[*] Address: 216 Langford Center, College Station, Texas 77843-3137. email: ergun@viz.tamu.edu. phone: +(409) 845-6599. Supported in part by the Texas A&M, Scholarly & Creative Activities Program.

that significant computational gains can be obtained by automatically adapting the complexity of a geometric model [9]. One common heuristic technique is to construct several versions of a model at various levels of details (LOD), i.e. a detailed mesh is used when the object is close to the viewer, and coarser approximations are substituted as the object recedes [9, 14]. Such LOD meshes can be computed automatically using mesh simplification application techniques [10, 18].

There has been extensive research on the development of effective LOD representations [9, 14, 10, 19, 18, 17]. Existing LOD representations can be classified as hierarchical and progressive. Hierarchical LOD methods are similar to wavelet representations [7], in which there exists only a distinct number of versions of a model. Despite their simplicity the most important problem with hierarchical LOD representations is that the instantaneous switching between LOD meshes may lead to perceptible "popping" artifacts. Moreover, hierarchical methods require a regular mesh structure (e.g., every face is quadrilateral and every vertex is valence 4; or every face is a triangle and every vertex is valence 6) [12].

Progressive meshes are developed to solve the problems of hierarchical method [18]. In progressive meshes, unlike hierarchical LOD representations, there is no distinct number of simplified shapes. Instead, the simplification is described as a continuous function. Because of this property a shape can gradually be simplified. Gradual simplification is very useful since a user can continuously zoom in the shape without seeing sudden changes in the shape of the 3D mesh. Moreover, the underlying mesh structure does not have to be regular. In progressive meshes [18], only triangular faces are allowed but there is no restriction over the valence of the vertices. Moreover, progressive representations naturally support progressive transmission, offers a concise encoding of polygonal mesh itself, permits selective refinement [18]. In short, progressive representations offer an efficient, lossless and continuous-resolution representation.

1.2 Topology Simplification

In order to talk about topology simplification we assume that initial meshes are 2-manifold and they stay 2-manifold in each stage of refirement process. This requirement is important for developing consistent framework. The requirement is not restrictive since any mesh that is given as a collection of polygon can be converted to a 2-manifold [8, 23].

Topological simplification refers the capability of gradually simplifying any given 2-manifold mesh to a simplest genus-0 surface, namely a tetrahedron. It is important to note that topological simplification does not require reducing of genus or combining surfaces (i.e. combining shells[1]) in every stage of refinement. During the refinement genus may increase but mesh will eventually simplify to a tetrahedron. However, the majority of topology changes will consist of either reducing of genus or combining surfaces.

As we have discussed earlier combining surfaces will be most common applications of topology simplification since in current graphics applications manifold meshes that consists of huge number of separate surfaces are extremely common. Genus reduction will probably not be a prevailing application in the near future since very high-genus manifold meshes are not very common in graphics applications. However, we predict that high genus meshes will eventually be common and widely available. Especially, our current research involves to develop a wide variety of methods to easily create very high genus meshes [4, 22, 5, 13].

It is very easy to demonstrate that topological simplification can provide efficient simplification. Consider a Sierpinsky tetrahedra that consists of infinite number (in practice, a huge number) of tetrahedra. A Sierpinsky tetrahedra cannot be simplified by any LOD scheme since its every element is already a simplest genus-0 mesh. However, common sense tells us a Sierpinsky tetrahedra should be able to be simplified into one tetrahedron. A simplified example is shown in Figure 1.

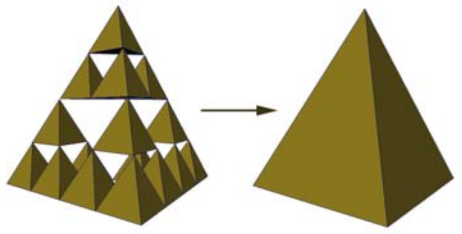

Figure 1. By allowing topology changes these 16 tetrahedra can be simplification into one tetrahedron.

1.3 Topological Simplification under Progessive Refinement

The topological simplification can be useful regardless of the choice of refinement approaches. Figures 2 and 3 show examples of topological simplification under two different refinement approaches. As these two figures illustrate, progressive refinement can provide a more natural looking simplification of topological features. However, topology simplification under a progressive refinement

[1]In topology, a surface is a connected 2-manifold [15]. In solid modeling literature, it is often called as *shell* [20, 16].

scheme introduces additional challenges that will not exist under hierarchical LOD schemes.

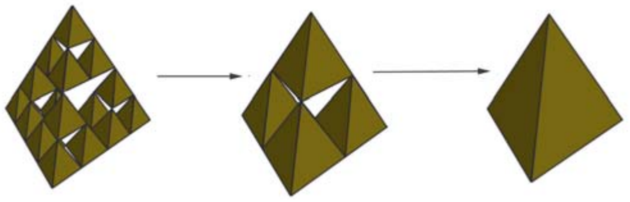

Figure 2. Hierarchical LOD with progressive refinement

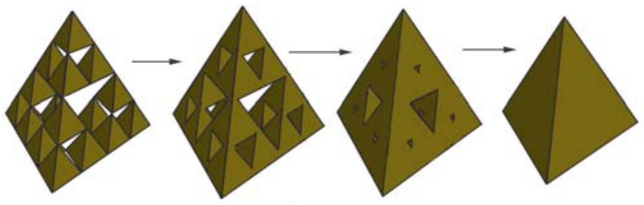

Figure 3. Progressive refinement with topology simplification

1.4 Problems with Commonly Used Topology Change Operators

The two most common topology change operators in computer graphics are CREATEPIPE and CUT (see Figure 4). These operators are also used in Hybrid meshes [12]. They are acceptable for hierarchical LOD schemes but cannot be used in a progressive refinement with topology simplification scheme. There are three problems with these operators for progressive refinement schemes.

1. Continuous change. With CREATEPIPE and CUT operators it is not possible to achieve gradual change. These operators requires many operations over the mesh that can create a very visible qualitative change.

2. Topological inconsistency. The implementations of these operators generally result in topological inconsistencies. These operators are generally implemented by two step processes.

   CUT operator case: the operator cuts off a pipe from the 2-manifold and leaves two open holes (i.e. manifold becomes manifold with boundary) in the first step. The second step cut seals the two open holes by two disks (which are the two new faces).

   CREATEPIPE operator case: General implementations of CREATEPIPE operator also follow a similar two step process. In the first step, operator deletes the faces f_1 and f_2, which results in a 2-manifold with boundary (with two “open” holes). In the second step, the operator creates a “pipe” (as a shape of prism) between the two holes and allows the pipe ends to “seal” the two holes. In some cases, instead of creating a pipe, vertices on pipe ends are simply merged.

   Such implementations that result in topological inconsistency (i.e. creating manifolds with boundaries as during the operation) are not acceptable for progressive refinement.

3. Flexibility. The concept behind CREATEPIPE and CUT operators is to create or delete a prism shaped pipe and the operators can reduce or increase the genus only by 1. However, the topology changes vary much more than simply creating or deleting prism shaped connections and we may sometime need to reduce genus more than 1.

   For instance, CREATEPIPE and CUT operators are not able to handle cases such as progressive refinement of Sierpinsky tetrahedron as shown in Figure 3. In this example, all the tetrahedra are uniformly converted to truncated tetrahedra (4 hexagons and 4 triangles) and tetrahedral shaped connections connect the truncated tetrahedra. Moreover, the holes at the centers of four tetrahedra are octahedral shape and closing these holes requires to reduce genus more than one.

Among these problems, topological inconsistency can be solved by using a different implementation approach as we will show in the next section. However, the other two cannot be solved.

2 Theoretical Framework

Our approach will be extension to progressive meshes. Thus, we will first introduce progressive mesh representations. Let $\mathcal{M}$ be an arbitrary triangular manifold mesh that can be represented by rotation system of a graph $\mathcal{G}(\mathcal{E}, \mathcal{V})$ where $\mathcal{E} = \{e_0, e_1, \ldots, e_m\}$ is a set of edges and and $\mathcal{V} = \{v_0, v_1, \ldots, v_k\}$ is a set of vertices.

A progressive representation (PM) of a manifold mesh $\mathcal{M}$ is stored as a much coarser power network $\mathcal{M}^0$ together with a sequence of n detailed records that indicate how to incrementally refine the coarse power network $\mathcal{M}^0$ exactly back into the original power network such that $\mathcal{M}^n = \mathcal{M}$. Each of these records will store the information associated with a mesh transformation operation. In progressive meshes this is VERTEXSPLIT operation.

The progressive representation (PM) of the mesh $\mathcal{M}$ thus defines a continuous sequence of power networks $\mathcal{M}^0, \mathcal{M}^1, \ldots, \mathcal{M}^n$ of increasing accuracy, from which the LOD approximation of any desired complexity can be efficiently retrieved. Moreover, geomorphs (continuous shape changes) can efficiently constructed between any two of these power networks.

To create continuous sequence of manifold meshes $\mathcal{M}^0, \mathcal{M}^1, \ldots, \mathcal{M}^n$, inverse of VERTEXSPLIT operation, EDGE COLLAPSE, is used. Since initial mesh is triangular, each EDGE COLLAPSE operation simplifies the mesh by eliminating two triangles, one edge and two vertices.

One of the great power of progressive mesh framework is that each direction (simplification and complication) requires only one type of operator. However, this theoretical framework creates two problems for topological simplification.

1. Vertex split and edge collapse cannot change the topology[2].
2. The current progreesive refinement framework only allows triangles but topology changes can require faces other than triangles.

To obtain topological simplification we have to use different operators that allow topology changes and non-triangular faces. The following section presents such a set of operators.

2.1 Minimal Operators

To provide topological simplification, instead of VERTEXSPLIT and EDGECOLLAPSE operators we propose to use CREATEVERTEX, DELETEVERTEX, INSERTEDGE and DELETEEDGE operators [4]. It has recently been proven that all and only manifold meshes can be created by these four operators [8]. Moreover, this set of operators is minimal and complete in the sense that it is necessary and sufficient for performing any homeomorphic and topological operation on any orientable 2-manifold meshes [8]. The properties of these operators can be summarized as follows:

1. CREATEVERTEX(v) is similar to Euler operation MVFS. It creates a new isolated vertex v and its corresponding face, which we simply call a *point-sphere*. This operator is essential to add a new surface component in the given 2-manifold and to create a surface in an empty manifold.
2. DELETEVERTEX(v) is similar to Euler operation KVFS. It deletes an isolated vertex and its corresponding point-sphere from the current 2-manifold. This operator is necessary to remove a surface component from the current manifold.
3. INSERTEDGE(c_1, c_2, e) inserts a new edge e to the mesh structure between two corners c_1 and c_2 (a *corner* is a subsequence of a face boundary walk consisting of two consecutive edges plus the vertex between them). If INSERTEDGE Inserts an edge between two corners of the same face, the new edge divides the face into two faces [2]. On the other hand, if INSERTEDGE inserts an edge between corners of two different faces, the new edge merges the two faces into one [2, 3].
4. DELETEEDGE(e) deletes an edge e from the current 2-manifold mesh. This is the inverse operator of INSERTEDGE. DELETEEDGE either merges two faces into a bigger face or splits a face (which forms a pipe) into two faces. An important function of the DELETEEDGE operator is to split a surface into two surfaces: If the DELETEEDGE operator on an edge $e = [v_1, v_2]$ completely disconnects the vertices v_1 and v_2 so that there is no further path connecting the vertices v_1 and v_2 in the mesh, then the surface that contains both vertices v_1 and v_2 is actually split into two disjointed surfaces, one containing vertex v_1 and the other containing vertex v_2.

These four operators are necessary and sufficient to implement all homoemorphic and topological changes on a given 2-manifold mesh [8]. For consistent topological change, it is important to apply only these operators. For instance, in order to create a pipe there is no need to cut off along the boundaries of the faces, which would result in a 2-manifold with boundary that will be topologically inconsistent. INSERTEDGE operator implicitly make the change without creating topological inconsistency.

Using a sequence of INSERTEDGE and DELETEEDGE operators we can easily obtain more familiar topology change operators CREATEPIPE and CUT as shown in Figure 4. As it is clearly shown in these examples there is no need to cut off along the boundaries of the faces for creating a pipe and there is no need to seal faces after cutting a pipe. The faces will be automatically cut and sealed by INSERTEDGE and DELETEEDGE operations. Therefore, during topology changes no inconsistent structure (i.e. manifold with boundary) will be created.

Besides solving topological inconsistencies using a series of INSERTEDGE and DELETEEDGE, there are additional challenges associated with topology changes. One of the reasons behind why the people try to avoid topology changes is that they can be non-intutive. There are two sources of this problem: (1) the same operator can both simplify and complicate the topology and (2) the same operator can both open (or close) holes and create (or delete) handles.

[2]Moreover, edge collapse can create non-manifold. Therefore, a lot of care is needed in order to ensure that the result is manifold [8].

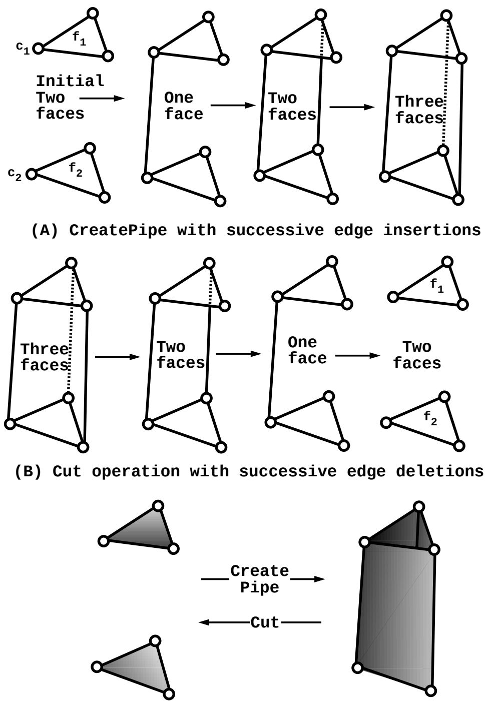

Figure 4. CREATEPIPE **and** CUT **operators.**

1. The same topology change operator can both simplify and complicate the topology. Following two example illustrates this problem.

    - The CREATEPIPE operation can both complicate and simplify the topology. As shown in Figure 5. In 5(A) CREATEPIPE operation combine two genus-0 surfaces creating one genus-0 surface by simplifying the topology. On the other hand, the same CREATEPIPE operation in 5(B) changes a genus-0 surface into a genus-1 surface by complicating topology.
    - The CUT operation can also both complicate and simplify the topology. As shown in Figure 6. In 6(A) CUT operation creates two genus-0 surfaces from one genus-0 surface by complicating the topology. On the other hand, the same CUT operation in 6(B) changes a genus-1 surface into a genus-0 surface by simplifying topology.

2. the same operator can both open (or close) holes and create (or delete) handles. Following two examples illustrate such cases.

    - The CREATEPIPE operation can create both handle or hole as shown in Figure 7.
    - The same operation can delete a handle or a hole as shown in Figure 8.

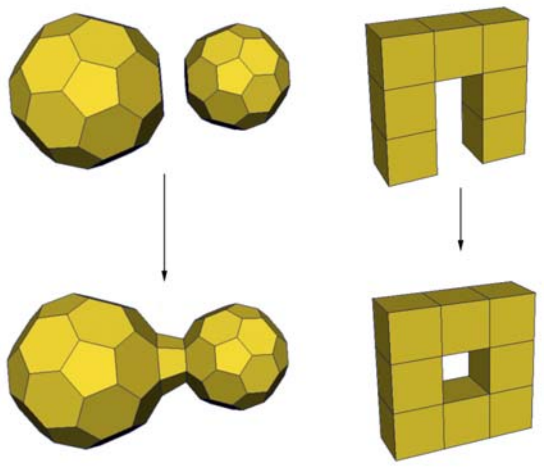

Figure 5. The CREATEPIPE **operation can both simplify and complicate the topology.**

Similar to the operators, CREATEPIPE and CUT, the operators INSERTEDGE and DELETEEDGE can also both simplify and complicate the topology. Moreover the same operator can both open (or close) holes and create (or delete) handles. The following diagram demonstrates the effect of these operators.

Operator	Genus	The number of surfaces
INSERTEDGE	Increases	Decreases
DELETEEDGE	Decreases	Increases

Operator	Holes	Handles
INSERTEDGE	Opens	Creates
DELETEEDGE	Closes	Deletes

Since the same operator can both complicate and simplify the topology, using topology change operators is not as straightforward as using vertex split and edge collapse operators. Therefore, it is neccessarry to be careful to guarantee simplification.

The above discussions clearly show that the operators INSERTEDGE and DELETEEDGE are inverse of each other. Once we obtained the operations for refinement process, to recover the initial mesh we will use the inverse operations. In other words, although the operations are not really intuitive, once the operations are determined, finding the inverse is straightforward. Thus, the problem is reduced to developing an algorithm that guarantees to simplifiy any 2-manifold mesh to a genus-0 surface.

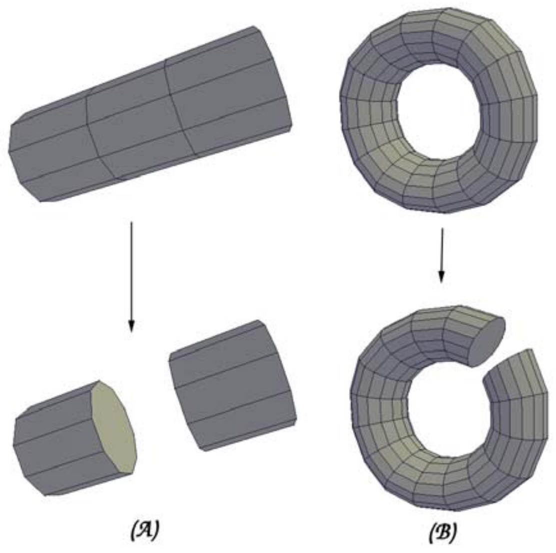

Figure 6. The CUT operation can both simplify and complicate the topology.

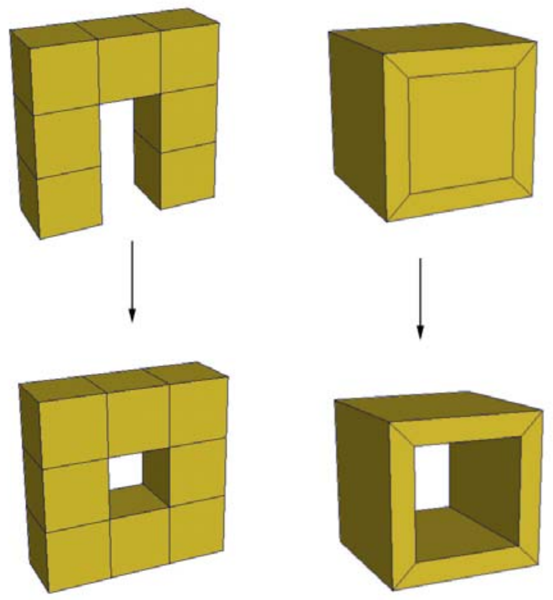

Figure 7. The CREATEPIPE operation can create both hole or handle.

3 Equidistance Surfaces to Simplify Topology

Since a manifold mesh is a closed and bounded shape equidistance surfaces to this manifold mesh will eventually simplifies to a genus-0 ball while distance is increasing regardless of the choice of distance function [6, 21, 11, 1]. The problem with equidistance surfaces is that they eventually become much larger than initial manifold shape. This problem can be solved by carefully scaling the shape.

Figures 9 and 10 illustrate the concept by using 2D representations. Note that Figure 10 clearly shows that genus can increase while distance increases. However, as it is clear in this example the genus will eventually simplify to 0.

One possible implementation of topological simplification with equidistance surfaces can be done by using offset surfaces [5]. As an example here we discuss how to create progressive meshes using offset surfaces. Let $\mathcal{A}^0$ be the initial mesh. We first need to compute a series of offset surfaces $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^n$. Here $\mathcal{A}^{k+1}$ will directly be computed from $\mathcal{A}^k$. If the offset surface does not have any self-intersection then $\mathcal{A}^{k+1}$ will simply be progressively refined version of $\mathcal{A}^k$. If there is a self intersection, we will remesh $\mathcal{A}^{k+1}$ by changing the topology. We will then apply the inverse of offset operation to $\mathcal{A}^{k+1}$ to scale the shape back to original size.

The scaling will generally be a non-linear deformation. First of all, all the vertices that are not effected by topology change or refinement should return back to their original position. If we do not move such vertices into their original positions, we would have to provide their positional changes in every step of progressive refinement. Of course, this is not desired. The initial positions of newly created and final positions of deleted vertices correspond the Morse points of distance function and can be computed. The interesting problem is to develop a non-linear scaling to compute the change in this positions. This non-linear scaling will be given by a deformation that minimizes the changes of positions all the other vertices.

We will continue this process until $\mathcal{A}^{k+1}$ becomes a genus-0 surface, preferably a tetrahedron. Then $\mathcal{M}^0 \leftarrow \mathcal{A}^n$ and inverse of the operations will be used to get initial mesh $\mathcal{M}^n$.

4 Conclusion and Future Work

In this paper, we present a theoretical framework for progressive refinement of manifold meshes with topological simplification. We demonstrate that topology changes are not intuitive and therefore a great deal of care is necessary for handling topological simplification. We illustrate non-intutive nature of topology changes with several examples. All the shapes shown in this paper are created using our interactive modeler [4]. We also show how we can use the non-intutive nature of the topology changes as an advantage and develop a theretical framework for progressive refinement with topological simplification. We propose a distance based concept for topologicl simplification and discussed one possible implementation using offset surfaces.

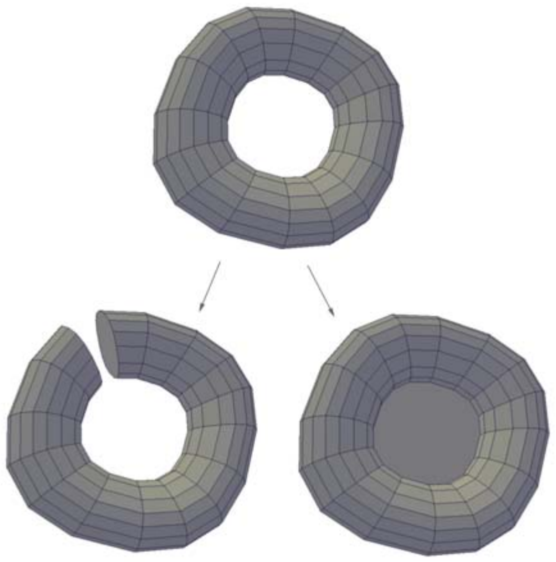

Figure 8. The CUT operation can delete either a handle or a hole.

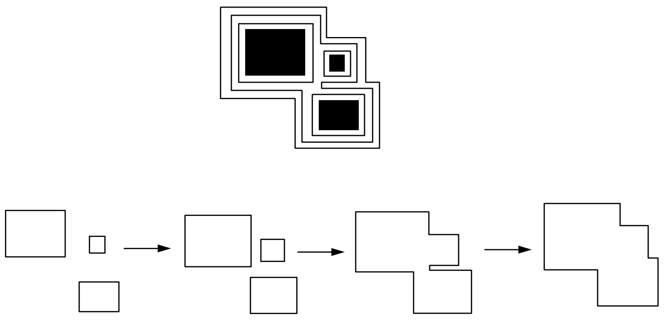

Figure 9. An 2D representation of combining surfaces by using equidistance shapes.

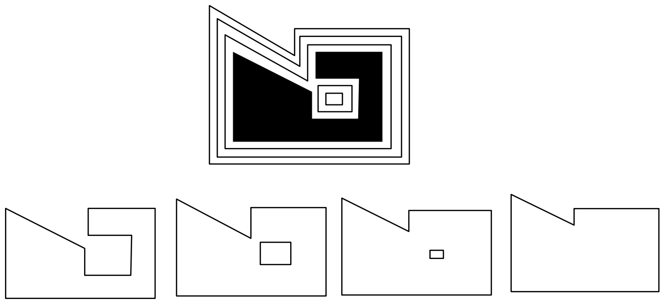

Figure 10. An 2D representation of genus simplification by using equidistance shapes.

References

[1] E. Akleman and J. Chen, Generalized Distance Functions, *Proceedings of Shape Modeling International'99*, (March 1999), pp. 72-79, Aizu, japan.

[2] E. Akleman and J. Chen, Guaranteeing the 2-manifold property for meshes with doubly linked face list, *International Journal of Shape Modeling 5*, (1999), pp. 149-177.

[3] E. Akleman, J. Chen, and V. Srinivasan, A new paradigm for changing topology during subdivision modeling, *Proceedings 8th Pacific Conference on Computer Graphics and Applications*, (PG'2000), (2000), pp. 192-201.

[4] E. Akleman, J. Chen and V. Srinivasan, An interactive system for modeling 2-manifold meshes, *Proceedings of International Conference on Shape Modeling and Applications 2002*, (SMI'02), (2002), pp. 43-50.

[5] E. Akleman V. Srinivasan, and J. Chen, Interactive Rind Modeling, Submitted to a conference. (See http://www-viz.tamu.edu/faculty/ergun/research/topology/ to get a copy).

[6] A. Bottino, W. Nuij and C. W. A. M. van Overveld, "How to Shrinkwrap through a Critical Point: An Algorithm for Adaptive Triangulation of Iso-Surfaces with Arbitrary Topology", *Proceedings of Implicit Surfaces'96*, pp. 53-72, October 1996.

[7] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin and W. Stuetzle, "Interactive Multiresolution Surface Viewing", *Computer Graphics SIGGRAPH*, pp 91-99, August, 1996.

[8] J. Chen and E. Akleman, Topologically robust mesh modeling: concepts, structures and operations, Submitted to a journal.(See http://www-viz.tamu.edu/faculty/ergun/research/topology/ to get a copy).

[9] J. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms", *Communications of ACM*, Vol. 19, No. 10, pp 547-554, October 1976.

[10] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks and W. Wright, "Simplification Envelopes", *Computer Graphics SIGGRAPH*, pp 119-128, August, 1996.

[11] S. F. Frisken, R. N. Perry, A. P. Rockwood and T. R. Jones, "Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics", *Computer Graphics*, No. 34, pp. 249-254, August 2000.

[12] I. Guskov, A. Kodakovsky, P. Schroder and W. Sweldens, “Hybrid Meshes: Multiresolution Using Regular and Irregular Refinement”, *Proceedings of ACM Symposium on Computational Geometry 2002*, pp. 264-272, June 2002.

[13] I. Guskov and Z. Woods, “Topological Noise Removal”, *Proceedings of Graphics Interface 2001*, pp. 19-26, June 2001.

[14] T. Funkhouser and C. Sequin “Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments”, *Computer Graphics SIGGRAPH*, pp 247-254, August, 1993.

[15] J. L. Gross and T. W. Tucker, *Topological Graph Theory*, (Wiley Interscience, New York, 1987).

[16] C. M. Hoffmann, *Geometric & Solid Modeling, An Introduction*, (Morgan Kaufman Publishers, Inc., San Mateo, Ca., 1989).

[17] H. Hoppe, “View Dependent Refinement of Progressive Meshes”, *Computer Graphics SIGGRAPH*, pp 189-198, August, 1997.

[18] H. Hoppe, “Progressive Meshes”, *Computer Graphics SIGGRAPH*, pp 99-108, August, 1996.

[19] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust and G. A. Turner, “Real-Time Continuous Level of Detail Rendering of Height Fields”, *Computer Graphics SIGGRAPH*, pp 109-118, August, 1996.

[20] M. MÄNTYLÄ, *An Introduction to Solid Modeling*, Computer Science Press, Rockville, MA, 1988.

[21] C. W. A. M. van Overveld and B. Wyvill, “Shrinkwrap: An Adaptive Algorithm for Polygonizing an Implicit Surface”, *The University of Calgary, Department of Computer Science, Research Report*, no. 93/514/19, 1993 .

[22] V. Srinivasan, E. Akleman and J. Chen, Interactive and user friendly construction of multi-segment curved handles, Submitted to a conference. (See http://www-viz.tamu.edu/faculty/ergun/research/topology/ to get a copy).

[23] V. Srinivasan, E. Akleman and J. Keyser, Topological Construction of 2-Manifold Meshes from Arbitrary Polygonal Data, Submitted to a conference.