A Minimal and Complete Set of Operators for the Development of Robust Manifold Mesh Modelers

Ergun Akleman [a,1,*], Jianer Chen [b], Vinod Srinivasan [c]

[a] *Visualization Laboratory, 216 Langford Center, Texas A&M University, College Station, Texas 77843-3137.*

[b] *Department of Computer Science, Texas A&M University, College Station, Texas 77843-3112.*

[c] *Visualization Laboratory, 216 Langford Center, Texas A&M University, College Station, Texas 77843-3137.*

Abstract

In this paper, we identify a minimal and complete set of fundamental operators, which is necessary and sufficient for performing all homeomorphic and topological operations on 2-manifold mesh structures. Efficient algorithms are developed for the implementation of these operators. We also developed a set of powerful, user-friendly, and effective operators at the level of user-interface. Using these operators, we have developed a prototype system for robust, interactive and user friendly modeling of orientable 2-manifold meshes. Users of our system can perform a large set of homeomorphic and topological changes with these user-interface level operators. Our system is topologically robust in the sense that users will never create invalid 2-manifold mesh structure with these operators.

In our system, the homeomorphic and topological surgery operations can be applied alternatively on 2-manifold meshes. With our system,users can blend surfaces, construct rinds and open holes on these rind shapes. With our system, the shapes that look like solid, non-manifold, or 2-manifold with boundary can be manipulated. The system also provides automatic texture mapping during topology changes.

* Corresponding Author. Email: ergun@viz.tamu.edu. phone: +(979) 845-6599. fax: +(979) 845-4491.

[1] Supported in part by Research Council of College of Architecture and Interdisciplinary Program of Texas A&M University.

1 Motivation and Introduction

Orientable 2-manifold meshes [15,17] can describe functional 3D shapes with arbitrary topological structure (i.e. having any number of holes, handles and "disconnected" surfaces) [16,14]. A 2-manifold mesh does not include unwanted mesh artifacts such as wrongly-oriented or missing polygons, cracks, and T-junctions [4,25]. These properties of 2-manifold shapes are essential for physical simulations. For instance, in order to simulate pouring water from a computer generated teapot shape, the artificial teapot must have a functional (not just apparent) spout. Similarly, for heat transfer, there should be no cracks, otherwise there can be an energy leak. In rendering, a model that has a crack or missing polygon can ruin the radiosity computation. In ray-tracing, a transparent shape with wrongly-oriented polygons and cracks can cause unwanted artifacts in the resulting image.

Orientable 2-manifold meshes are also essential for subdivision schemes since most subdivision algorithms such as Catmull-Clark and Doo-Sabin require that the local topological region must correspond to a simple disk (topologically) [8,12,30]. This implies that the underlying structure must be a valid 2-manifold. Since the quality and topology of the smooth surface resulting from subdivision depends greatly on the initial control mesh, theoretical assurance of the quality of initial control meshes is extremely important. In other words, the process of obtaining the initial control mesh must be robust and guarantee a valid 2-manifold structure.

This paper presents a prototype system for robust, interactive and user-friendly modeling of 2-manifold meshes based on topological graph theory. The main contributions of this paper is the following.

- *Minimal and complete set of operators.* We have identified four operators that can create any 2-manifold mesh. These operators can be implemented very efficiently over our Dubly Linked Face List (DLFL) data structure [1]. Once these operators are implemented, the development of a 2-manifold modeling system with high-level operators becomes extremely easy.
- *User-Interface operators.* We have also developed and implemented a set of user-friendly high-level operators based on the minimal operator set. By using these high-level operators, users can intuitively perform topological and homeomorphic surgery operations on a given 2-manifold mesh structure.
- *New design methods.* One of the most important contributions of this paper is the introduction of two new design methods, rind modeling and surface blending. rind modeling method is useful to design shapes such as teapots. Surface blending helps to create shapes that resemble blobby shapes that are generally created by implicit methods.

We have recently introduced topological graph theory to computer graphics for effective and robust modeling of 2-manifold meshes [1]. Based on topological graph theory, we have identified two operators InsertEdge and DeleteEdge to change mesh structure and manifold topology. In this paper, we combine these two operators with two fundamental Euler operators, MVFS and KVFS, which we call CreateVertex and DeleteVertex respectively. We demonstrate that these four operators constitute a necessary and sufficient minimal-set upon which all homeomorphic and topological operators can be implemented.

Based on these four operators, we have developed various high-level user interface operators and developed an interactive 2-manifold modeling system to demonstrate robustness and usability of our approach. Using the system, we have created various polygonal meshes that would be extremely difficult to model interactively without our system.

In our prototype system, users can make topological changes such as adding and deleting handles or holes, combining two surfaces into one and separating one surface into two. They also can perform the useful homeomorphic operations of vertex insertion or corner cutting subdivision schemes. As we have mentioned earlier, the algorithms of these schemes are also implemented based upon only the minimal operator set.

We have also introduced two new design methods, rind modeling and surface blending. Figures 1, 2 and 3 represent examples of models created by *rind modeling* method. The inspiration for the model shown in Figure 1 comes from chinese sculptures consisting of a set of nested rotatable balls. The actual sculptures can have up to 16 nested balls. Our version consists of three surfaces with genera 31, 31 and 41, respectively. Figure 2 shows another nested 2-manifold. For this model, we were inspired by a certain type of Indian sculpture, in which a small model of an animal is seen from the holes of a larger model of the same animal.

Creating holes and handles is not only useful for aesthetic purposes. In fact, the holes and handles are essential to construct functional models. The teapot shown in Figure 3 represents an example of a functional model created by our rind modeling method. As it can be seen from an x-ray image, this teapot has a *real* (not just a "look-like") hole to let the water pour from the spout. Because of the hole in the spout, this teapot can be used in physical simulations. The hole in the spout and the handle are designed in our system starting from a few number of rectangular prisms.

An additional benefit of topology changes during 2-manifold polygonal mesh modeling is the automatic creation of texture coordinates. With our system, it is possible to map a rectangular texture over a mesh of arbitrary topological

Fig. 1. 2-Manifolds with high genus.

structure by constructing it from a sphere or from a torus with given texture coordinates. The shape in Figure 4, which is created by our surface blending approach, shows an example of texture mapping when topology changes. Note that surface blending demonstrates that our system can be used to create bloby shapes which are similar to implicit bloby shapes [6,29] with polygonal meshes.

2 Background

The most commonly used operators to obtain meshes are the set operations of solid modeling. Unfortunately, in current practice, solid modeling allows non-manifold representations in order to simplify the algorithms [21] and the existing data structures are specifically developed to represent non-manifold surfaces [22]. It is worthwhile to note that although the data structures, such as extended winged-edge, can handle some non-manifold surfaces [5,27,22], data structures that can support a wider range of non-manifold surfaces have been investigated. Examples of such work are Weiler's *radial-edge* structure [28], Karasick's *star-edge* structure [23], and Vanecek's *edge-based* data structure [26].

One of the oldest formalized data structures that is designed to support manifold meshes is the *half-edge* representation [24]. Mantyla also suggested Euler operators [24] to change the topology and the structure of meshes. Unfortunately, half-edge structures can accept some specific non-manifold surfaces [11]. Guibas and Stolfi introduced edge algebra, the *quad-edge* data structure and make-edge and splice operators[19].

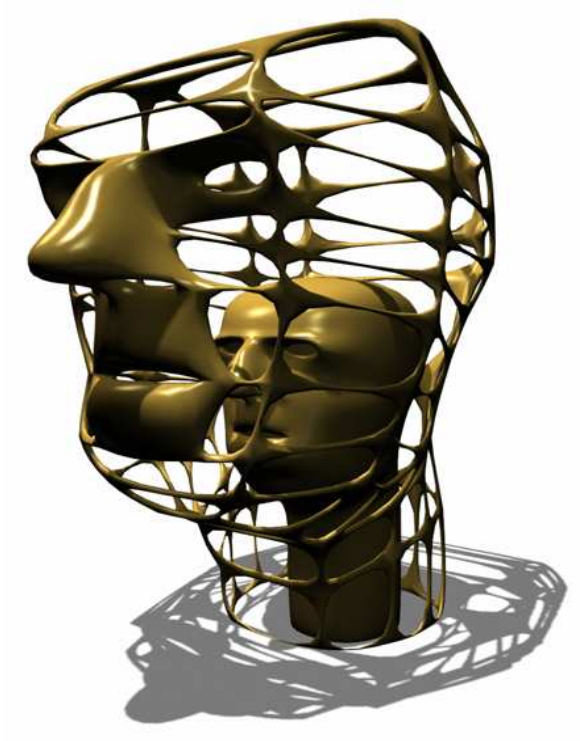

Fig. 2. An example of 2-manifold that is designed by our system.

Fig. 3. A teapot created by using our system.

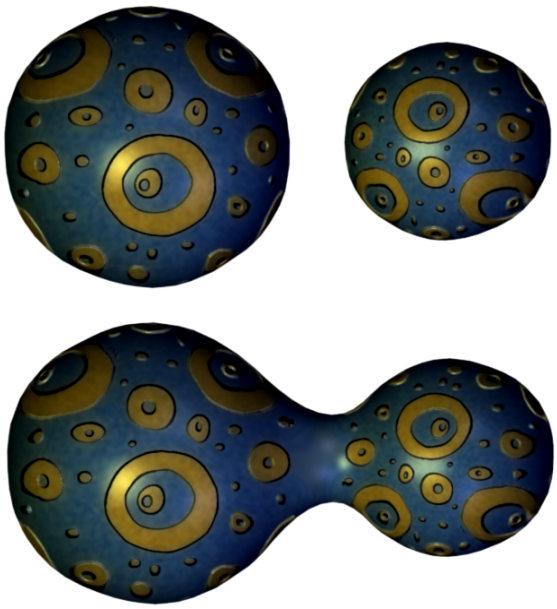

Fig. 4. An example of how textures are mapped when topology changes.

In solid modeling, it has been known that it is possible to make the internal representation of objects valid orientable 2-manifold structures while allowing their corresponding geometric shapes to look like non-manifolds [21]. The main concern is the consistency of the topological structures (which are valid 2-manifolds) and the geometric shapes (which look non-manifolds). In particular, with the non-manifold looking geometric shapes, ensuring that their internal representations are valid orientable 2-manifold structures is considered to be a difficult problem. Edge algebra gives tools to check the validity of 2-manifold structures [19]. However, operations such as subdivision and checking the validity of edge algebra implemented on quad edge structures can be very time consuming [11].

A relatively obscure mathematical theory that was introduced almost 100 years ago and known by only a handful of graph topologists gives a perfect and extremely simple solution to this problem. The classical theory of *graph rotation systems* originated from the study of graph embeddings [20] and it is implicitly due to Heffter [20] who used it in Poincare dual form. A graph embedding in an orientable 2-manifold corresponds to an obvious rotation system, namely, the one in which the rotation at each vertex is consistent with the cyclic order of the neighboring vertices in the embedding. Edmonds [13] was the first to call attention explicitly to studying rotation systems of a graph. He showed that every rotation system of a graph gives a *unique* orientable 2-manifold and the corresponding orientable 2-manifold is constructible. Moreover, for any orientable 2-manifold there always exists a graph rotation system that corresponds to an embedding on the 2-manifold

[9]. Thus, every orientable 2-manifold can be represented by a rotation system of a graph, and every rotation system of a graph corresponds to a valid 2-manifold. In consequence, the presentation of graph rotation systems always guarantees topological consistency.

We have recently developed an efficient data structure to implement graph rotation systems. Our data structure called *Doubly Linked Face List* (DLFL) [10,1] not only supports efficient computations on orientable 2-manifold meshes, but also always guarantees topological consistency, i.e. it always gives a valid 2-manifold mesh. Using DLFL we can efficiently check the validity of orientable 2-manifold structures. We also introduced new topological graph operators. The biggest advantage of our operators is that they are extremely simple while always guaranteeing topological consistency.

3 Minimal Set of Operators for Modeling Orientable 2-Manifold Meshes

Our InsertEdge and DeleteEdge operators can create both topological and homeomorphic changes. We noticed that when we combine these operators with two operators that are similar to Euler operators MVFS and KVFS [24], we obtain a minimal set of operators. MVFS creates a single skeletal primitive that consists of one vertex and one face without an edge (which we call *point-sphere*) introduced by Mantyla [24]. Point-sphere and its DLFL representation are shown Figure 5. The combination of the following four operators can uniquely describe every manifold preserving operator.

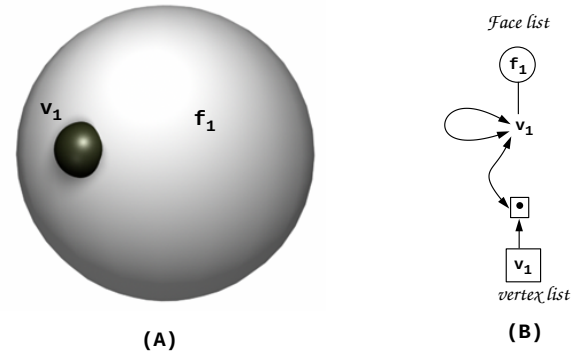

Fig. 5. A point-sphere and its DLFL representation.

(1) CreateVertex(v) creates a new isolated vertex v and its corresponding point-sphere. This operator is similar to Euler operator MVFS [2] and effectively adds a new surface component to the current 2-manifold.
(2) DeleteVertex(v) deletes an isolated vertex and its corresponding point-sphere from the current 2-manifold. This operator is similar to Euler operator KVFS and is used to effectively 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).
(4) DeleteEdge(e) deletes an edge e from the current 2-manifold mesh represented by the DLFL structure. This is the inverse operator of InsertEdge.

We argue that the above set of operators is minimal and complete in the sense that it is necessary and sufficient for performing any homeomorphic and topological operator on orientable 2-manifold meshes.

The CreateVertex operator is essential in the initial stage of manipulating 2-manifolds and creates a new surface component in the given 2-manifold. In particular, it is necessary when a new surface component is created in an empty manifold. Similarly, the DeleteVertex operator is convenient for "cleaning up" a manifold, and is a necessary operator when a surface component needs to be completely removed from the 2-manifold.

We have introduced the operators InsertEdge and DeleteEdge earlier [1]. If InsertEdge Inserts an edge between two corners of the same face, the new edge divides the face into two faces, as shown in Figure 6. On the other hand, if InsertEdge inserts an edge between corners of two different faces (this includes the situation in which an endpoint or both endpoints of the new edge correspond to point-spheres), the new edge merges the two faces into one. This situation is shown in Figure 7. The InsertEdge is the only operator among the four basic operators that can increase the genus of a surface.

In case the operator InsertEdge is inserting an edge e to corners of two different faces f_1 and f_2, there is an interesting and intuitive interpretation of the operation as follows: the operation can be decomposed into two steps. In the first step, we cut off along the boundaries of the faces f_1 and f_2, which results in a 2-manifold with two "open" holes, then the second step runs a new "pipe" between the two holes and allows the pipe ends to "seal" the two holes. An illustration is shown in Figure 8.

[2] It is not Euler operator since our data structure DLFL unlike half-edge data structure [24] does not keep track of shells, which are connected manifolds.

A special case for the InsertEdge operator is that an end of the new edge is an isolated vertex. In this case, the corresponding point-sphere is "merged" into the 2-manifold that contains the other end of the new edge. The isolated vertex now becomes a valence-1 vertex.

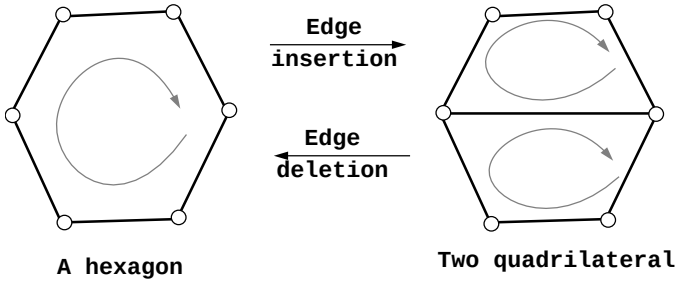

Fig. 6. Inserting an edge between two corners of the same face divides it into two faces and deleting an edge between two faces merges the two faces into one.

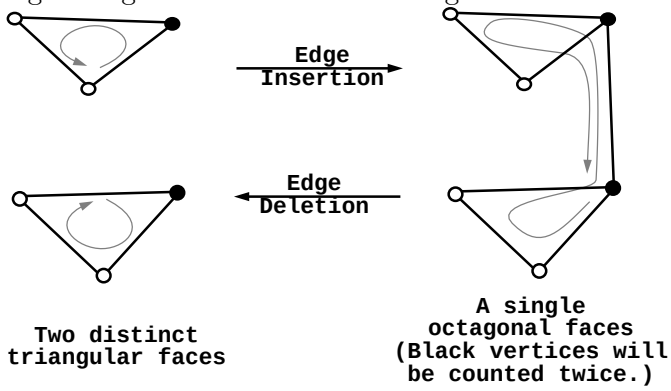

Fig. 7. Inserting an edge between two different faces merges the two faces and deleting an edge with both sides in the same face splits the face into two.

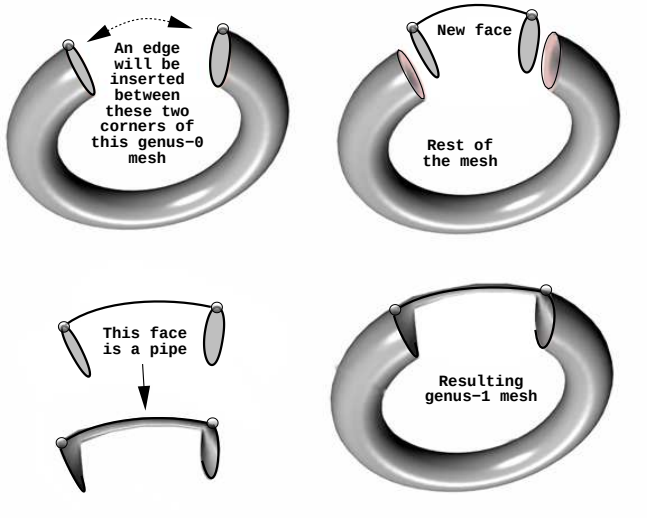

Fig. 8. A multiple-step representative illustration of creating handle by inserting an edge. (Note that cutting off a handle can be considered the reverse order of this process).

DeleteEdge is the inverse operator of the InsertEdge. As shown in Figure 6 and Figure 7, DeleteEdge either merges two faces into a bigger face or splits a face (which forms a pipe) into two faces. Similarly, when DeleteEdge splits a face into two faces (as shown in Figure 7), the situation can be described by a 2-step process in which the first step cuts off a pipe from the 2-manifold and leaves two open holes. The second step seals the two open holes by two disks (which are the two new 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.

A special case of this situation is when an end of the edge e is a valence-1 vertex. In this case, an isolated vertex, thus an invisible small point-sphere is created. This special case has been dealt with carefully in our implementation.

4 User-Interface Level Operators

The minimal set of fundamental operators is built at the implementation level of the extended DLFL structure, and may not be effectively visible at the user interface level because current computer graphics hardware do not provide adequate visual support. For instance, the shape of a handle or a hole that is created by the InsertEdge operator may not be clearly visible in current graphics hardware which, if used at user-interface level, would make it difficult to apply subsequent operators. Another example is the operators CreateVertex and DeleteVertex. These operators are dealing with infinitely small point-spheres that are difficult for users to detect.

We have developed a set of powerful and effective operators at a high level of user-interface, based on the minimal set of fundamental operators. Using these user-interface level operators, users can perform a large set of homeomorphic and topological operators effectively on a given 2-manifold structure. Moreover, since the operators are constructed based on the basic minimal set of fundamental operators, the user-interface system is very robust and topologically secured, in that users will never introduce invalid topological structures (i.e., non-2-manifold structures) to a given 2-manifold structure.

We present a number of examples of user-interface level operators implemented in our extended system as follows.

RemoveEdge(e) permits users to remove an edge, and in case isolated vertices are resulted after removing the edge, the resulted isolated vertices are also removed. This operator prevents users from unintentionally leaving unwanted point-spheres in edge deletion operations. Clearly, the RemoveEdge can be easily implemented using the fundamental operators DeleteEdge and DeleteVertex.

As we have discussed in the previous section, when the fundamental operator InsertEdge inserts an edge between corners of two different faces, the geometric effect in image is that a very small new pipe is run between the two faces, which merges the two faces into one. This results in an infinitely thin pipe. Because of the limit of the current graphics hardware, the resulting unconventional face structure is also difficult for users to understand when they apply subsequent operations on the mesh structure.

To provide users with more "natural" topological operators that produce high quality handles and holes, we developed a CreatePipe operator at the user-interface level. When two different faces are specified for running a pipe between them, the CreatePipe operator runs a much thicker pipe between the two faces, as shown in Figure 9. Implementation of this operator based on the fundamental operators is also straightforward: instead of a single call to the fundamental operator InsertEdge that inserts a single new edge between corners of the two faces, we make multiple calls to InsertEdge that also inserts additional edges between the corresponding corners of the faces. Note that another advantage of the operator CreatePipe is that the size of the handle or the hole can actually be controlled; it is simply decided by the corresponding face sizes.

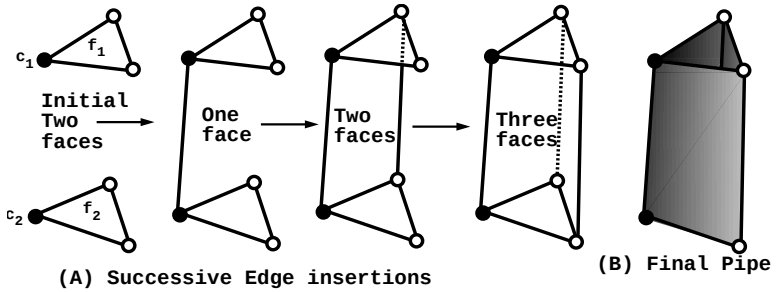

Fig. 9. Creation of a high quality handle/hole by inserting a set of edges.

We also found it useful to provide a SubdivideEdge operator to users that simply subdivides an edge into two edges by creating a new valence-2 vertex at the middle of the edge. This operator can be implemented by the fundamental operators by one DeleteEdge, one CreateVertex, and two InsertEdge operators. This operator allows users to create new vertices in a 2-manifold mesh without changing its topological structure, which is heavily used in the implementation of various subdivision schemes.

Various CreatePolyhedra operator are also developed and implemented. Using the CreatePolyhedra operators, users are able to create regular polyhedra, such as cubes, tetrahedra, octahedra, icosahedra, dodecahedra and polyhedral torii.

Various Subdivision schemes are also developed and implemented. Using the Subdivision schemes, users can apply a variety of subdivision schemes to a 2-manifold mesh, including a number of vertex-insertion schemes, such as Catmull-Clark scheme [8] and, a number of corner-cutting schemes, such as Doo-Sabin scheme [12].

5 Prototype System

Our system is implemented in C++ and fltk. All our interactive examples are running on SGI-O2. Our interactive renderer supports silhouette, wireframe, and texture mapping. We use a non-photorealistic lighting model for rendering since it does not leave any part of the object completely in shadow [18]. All the images that show wireframe, such as the ones in Figure 11, are screen snapshots from our system.

To create high-quality photorealistic images, we use a commercial modeling animation system, Maya. The images such as those in Figures 1, 2, 3, and 4 are rendered by using Maya's renderer. The system supports only topology and homeomorphic operations we discussed in the paper. It does not support geometry changes, i.e., it does not let users change the positions of vertices.

Our system can read and write in obj file format and the interaction with Maya is done in obj format. One limitation with this approach is that some of the 2-manifolds we have constructed in our system cannot be represented by obj format. However, the shapes after subdivision become more classical polygonal meshes and can be converted in obj file format.

Although it is a polygonal surface modeler, with our system users can create shapes that look like they have been created by a different type of modeler, such as solid or implicit modelers.

One of the unexpected results from user-interaction with our system is that users may feel that they are dealing with solid shapes. While using the system, some users perceive polygons as visible portions of solid bricks. Opening a hole gives the impression of removing a brick, and creating a handle is like putting a pipe or brick between two polygons. We show why this impression is formed by the examples given in Figures 10 and 11.

In the first example, we start with a surface that looks like a wall made up of 9 cubes (see Figure 10(A)). When we delete all the edges in the middle row, the wall automatically separates into two separated surfaces as shown in (C). This separation supports the feeling that the wall is actually solid. By the insertion of edges as shown in (D), the illusion of individual cubes can be obtained.

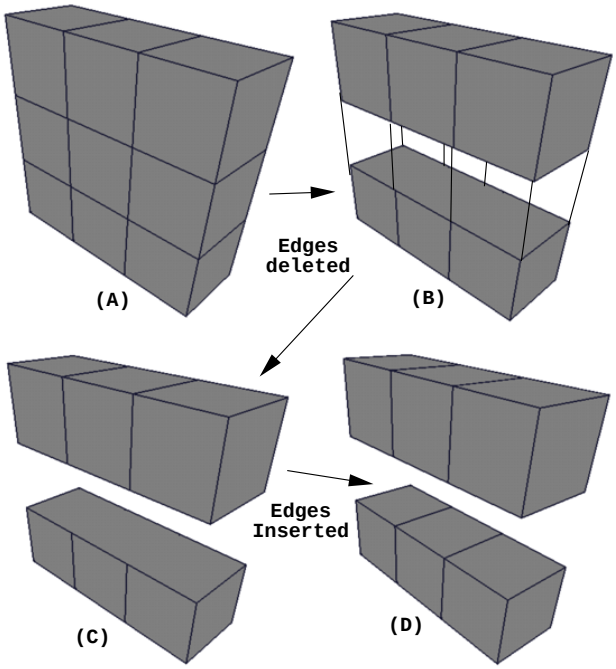

Fig. 10. Dividing a simple 2-manifold surface that looks like a wall made up of 9 cubes into two separate surfaces.

Figure 11 demonstrates another example of solid-like and non-manifold looking shape. In this example starting from the same wall, we first open a hole in the wall by the CreatePipe operator. The impact of this operation looks like the removal of a solid cube from the solid wall. Now if we remove all the edges at the upper left corner of the wall and insert a new edge to separate the newly created non-planar hexagonal face (see Figure 11(E)), the resulting shape looks like a non-manifold because of the infinitely thin space where the new edge is inserted. This, in fact, shows exactly another power of our system that can represent non-manifold looking shapes using valid 2-manifold structures: in the DLFL representation, the newly inserted edge is different from the edge that connects the same two vertices through the hole, although geometrically, the two edges have exactly the same position. This conclusion can be verified by applying a Subdivision operation on the shape (as shown in Figure 11(E)), which is only applicable to 2-manifold structures. From the resulting shape, we can see that because of the smoothing, the infinitely thin segment, which has two different edges on its two sides, becomes thicker, showing the resulting surface a topologically valid torus.

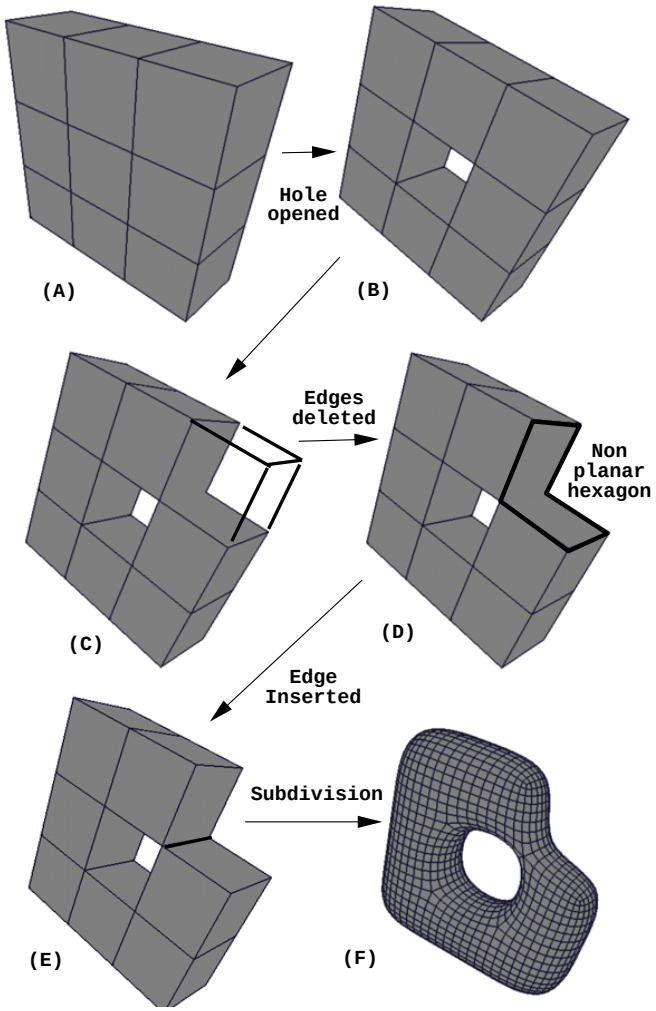

Fig. 11. Modeling a simple 2-manifold that looks like a non-manifold.

A straightforward extension of our method is automatic texture mapping during topological changes. The only necessary extension is to include texture coordinates to vertex data. With this extension our 2-manifolds live in a 5D instead of 3D. An example of automatic texture mapping is shown in Figure 4.

Careful attention is necessary during the subdivision. Our experience suggests that if the Catmull-Clark subdivision [8] is applied to texture coordinates, textures appear to move. Therefore, instead of using Cattmul-Clark, we have used an interpolation scheme for texture coordinates.

6 New Design Methods

This section introduces two new design methods that can be used with our system, rind modeling and surface blending. We present a number of examples

to illustrate these methods.

6.1 Rind Modeling

By using our system it is easy to create shells that are similar to a coconut husk and open holes on this husk. The procedure is as follows: (1) Duplicate the mesh and scale the new mesh so that it will completely be inside of the first mesh (if the object is not convex or star, it may be necessary to furhter move the positions of some vertices). This operator creates two nested surfaces. (2) Reverse the normals of the smaller second mesh. This operator changes the inside and outside of a 2-manifold mesh by changing the rotation orders of faces. (3) Connect any two corresponding faces by using the CreatePipe operator. This operator connects two surfaces and makes them a single surface. Then, additional holes can be created by opening more holes using the CreatePipe operator.

Figure 12 demonstrates two different models created by our rind modeling. The initial mesh is two nested truncated icosahedra shown in (A). (The inner (unseen) truncated icosahedron's normals are reversed). (B) shows 12 pentagonal holes. Note that although there are 12 pentagonal holes, the shape in (B) has genus 11 – the first CreatePipe operator does not increase genus since it actually connects two genus-0 surfaces. (C) shows a genus-41 surface that is obtained by application of a corner-cutting scheme [3] and then opening rectangular holes. The resulting shape after subdivision is shown in (D). Figure (E) shows a genus-31 surface that is obtained by opening a hole for each of the pentagonal and hexagonal faces. Note that this surface also looks like non-manifold. However, after subdivision, as shown in (F), the real structure becomes visible. The shapes in (F) and (D) are used in the creation of the object shown in Figure 1. (F) is the outer surface and (D) is the inner surface. The one in between created by first applying corner-cutting subdivision scheme to the surfaces in (A) and then opening pentagonal and hexagonal holes.

The shapes in Figure 2 were also created as rinds. In that case, first an initial face was created and duplicated. Then, we created a mask and opened holes in the positions of the eyes, mouth and nostrils. After we finished the mask, the outer shell was designed by opening many holes. The teapot in Figure 3 was also created using the same method.

We would also like to point out that surfaces that look like *2-manifold with boundary* can be created with a modified version of this method: 1) duplicate the mesh; 2) reverse the normals of the new mesh; and 3) get rid of some faces by opening holes.

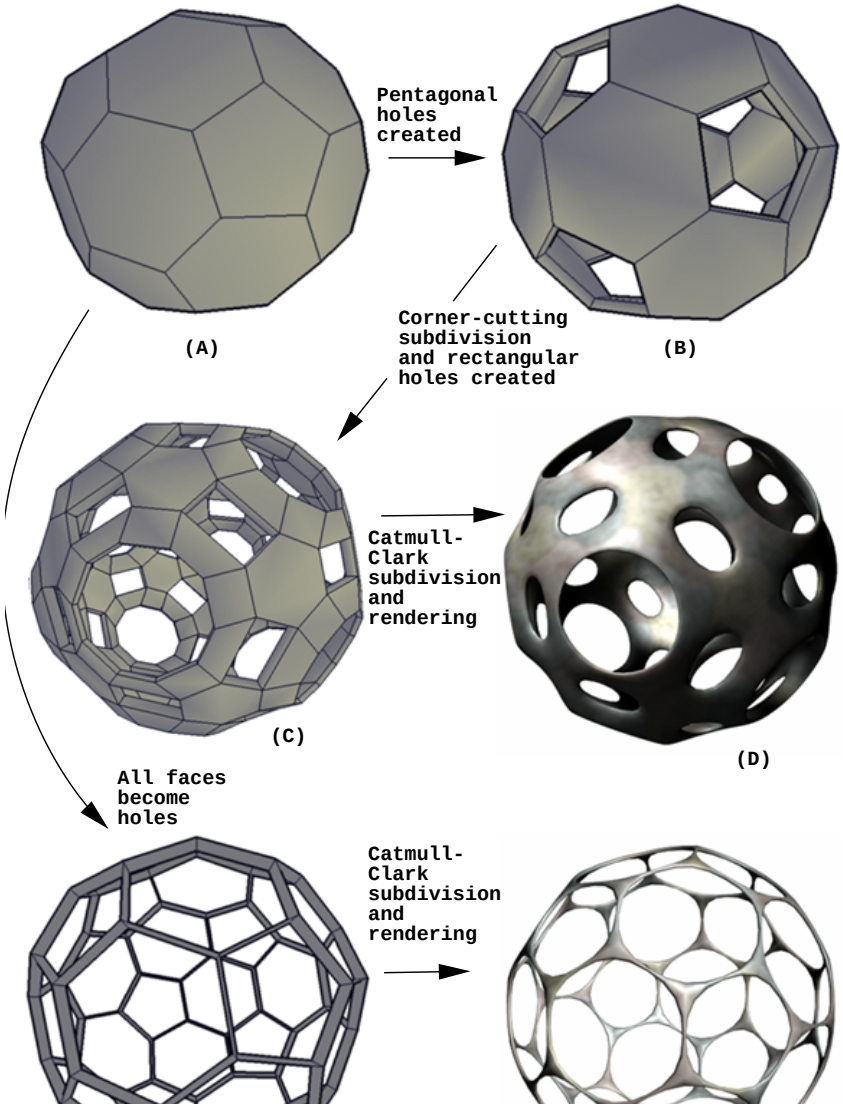

Fig. 12. Example of modeling a rind shape.

6.2 Surface Blending

Users of our system can also create implicit-like blending between surfaces by the following procedure: first connect the surfaces by connecting two nearest faces by the CreatePipe operator, then apply a subdivision scheme. If necessary, the faces can be remeshed by adding, deleting and subdividing edges to achieve a blended look before applying subdivision.

Figure 4 demonstrate implicit-like blending. In this example, we start with

two truncated icosahedra [3] and connect the two nearest pentagons by the CreatePipe operator. Then, we apply Catmull-Clark subdivision [8] to both initial two truncated icosahedra and connected surface to get two images in Figure 4.

Figure 13 shows another example of implicit-like blending: tree-like branching shapes [7]. This case is more complicated than the previous example because the four surfaces must be connected. By connecting two surfaces each time and remeshing the resulting structure we eventually created the surface shown in (B). Then, we achieved implicit-look by applying Catmull-Clark subdivision scheme. A high quality rendering is shown in (D).

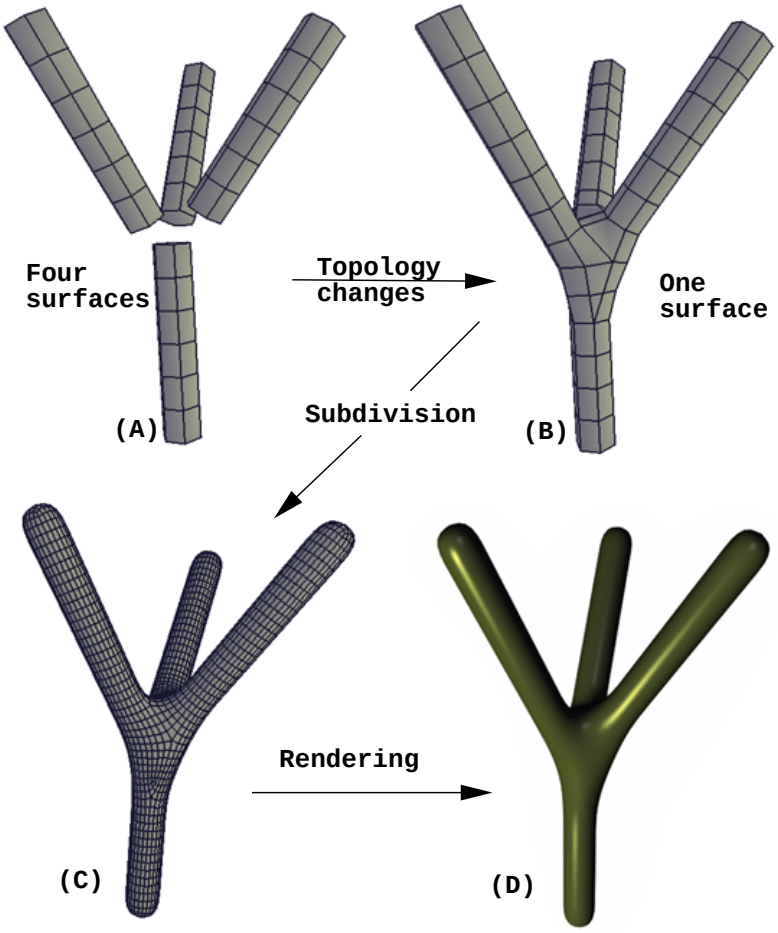

Fig. 13. Implicit-like modeling.

[3] One example of truncated icosahedron is shown in Figure 12(A). This polyhedron consists of 12 pentagons and 20 hexagons similar to soccerballs and it is the basic structure behind the geodesic dome.

7 Conclusion and Future Work

Note that based on our minimal set of operators it is easy to expand the capabilities of the system. In fact, homeomorphic operations that are applicable in our extended class are not just limited to regular subdivision schemes, any homeomorphic operation over 2-manifold shapes can be effectively implemented using our minimal-set.

In this paper, we have introduced new topological entities for effectively manipulating 2-manifold mesh structures. We have identified a minimal set of fundamental operators, which is necessary and sufficient for performing all homeomorphic and topological operators on 2-manifold mesh structures. We developed a system in which extremely efficient algorithms for these operators are implemented. A set of powerful, user-friendly, and effective operators has also been developed at the level of user-interface, based on the fundamental operators. Users of our system can perform a large set of homeomorphic and topological changes with these user-interface level operators. Moreover, because our system is based on an extended DLFL structure that always guarantees a valid 2-manifold structure, it is topologically robust in the sense that users will never create invalid 2-manifold mesh structure with these operators.

We have also introduced a number of methods, e.g., surface blending and rind modeling, to create shapes with the new modeler. By using these methods, we have constructed a wide variety of shapes that cannot easily be constructed otherwise.

Our work has shown that the system is also powerful and practical in manipulating non-manifold looking shape structures. We have presented examples, in which many shapes that are not conventionally considered to have a 2-manifold representation can have 2-manifold representations and can be constructed by our modeler.

We are planning to introduce new methods and develop high-level operators for these methods. Such high-level operators can be useful not only in modeling but also in other fields of computer graphics. For instance, blending can be used in physical based modeling to represent interacting liquid particles. Another possible usage of our approach is mesh compression. Since our extended structure provides DeleteEdge and DeleteVertex operators to simplify the topology of a given mesh, these topology simplifications can be used to obtain better and more efficient simplifications of meshes. Moreover, our approach can also be used to develop algorithms to provide morphs between polygonal meshes of different genii.

8 Acknowledgments

We are grateful to Donald H. House, David Romei and Mark Clayton for their valuable comments. We would also like to thank Sajan Skaria for initial facial model that is used to create the shape shown in Figure 2. Glen Vigus and Christina Haaser helped us to edit the video that shows an interactive session.

References

[1] E. Akleman and J. Chen, "Guaranteeing the 2-Manifold Property for meshes with Doubly Linked Face List", *International Journal of Shape Modeling* Volume 5, No 2, pp. 149-177.

[2] E. Akleman, J. Chen, and V. Srinivasan, "A New Paradigm for Changing Topology During Subdivision Modeling," *Pacific Graphics 2000*, October 2000, pp. 192-201.

[3] E. Akleman, J. Chen, F. Eryoldas and V. Srinivasan, "Handle and Hole Improvement with a New Corner Cutting Scheme with Tension," *Shape Modeling 2001*, May 2001, pp. 183-192.

[4] G. Barequet and S. Kumar, "Repairing CAD models", in *Proceedings of IEEE Visualization'97*, (October 1997) pp. 363-370.

[5] B. J. Baumgart, "Winged-edge polyhedron representation", Technical Report CS-320, Stanford University, 1972.

[6] J. I. Blinn, "A Generalization of Algebraic Surface Drawing", *ACM Transaction on Graphics,* vol 1, no. 3, pp. 235-256, 1982.

[7] J. Bloomenthal, "Modeling the Mighty Apple", *Computer Graphics,* vol 19, no. 3, pp. 305-311, 1985.

[8] E. Catmull and J. Clark, "Recursively generated B-spline surfaces on arbitrary topological meshes", *Computer Aided Design*, **10** (September 1978) 350-355.

[9] J. Chen, D. Archdeacon, and J. Gross, "Maximum genus and connectivity", *Discrete Mathematics*, **149** (1996) 19-29.

[10] J. Chen, "Algorithmic graph embeddings", *Theoretical Computer Science*, **181** (1997) 247-266.

[11] J. Chen and E. Akleman, "Topologically Robust Mesh Modeling: Concepts, Data Structures and Operations", *In Preparation.*

[12] D. Doo and M. Sabin, "Behaviour of recursive subdivision surfaces near extraordinary points", *Computer Aided Design*, **10** (September 1978) 356-360.

[13] J. Edmonds, "A Combinatorial Representation for Polyhedral Surfaces", *Notices American Mathematics Society*, **7** (1960) 646.

[14] H. Ferguson, A. Rockwood and J. Cox, "Topological Design of Sculptured Surfaces", *Computer Graphics*, **26** (August 1992) 149-156.

[15] P. A. Firby and C. F. Gardiner, *Surface Topology*, (John Wiley & Sons, 1982).

[16] A. T. Fomenko and T. L. Kunii, *Topological Modeling for Visualization*, (Springer-Verlag, New York, 1997).

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

[18] A. Gooch, B. Gooch, P. Shirley and E. Cohen, "A Non-Photorealistic Lighting Model For Automatic Technical Illustration", *Computer Graphics,* vol 32, no. 3, pp. 447-452, 1998.

[19] L. Guibas, J. Stolfi, "Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams", *ACM Transaction on Graphics*, **4** (1985) 74-123.

[20] L. Heffter, "Uber das Problem der Nachbargebiete", *Math. Annalen*, **38** (1891) 477-508.

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

[22] C. M. Hoffmann and G. Vanecek, "Fundamental techniques for geometric and solid modeling", *Manufacturing and Automation Systems: Techniques and Technologies*, **48** (1990) 347-356.

[23] M. Karasick, "On the representation and manipulation of rigid solids", Ph.D. Thesis, McGill University, 1988.

[24] M. Mantyla, *An Introduction to Solid Modeling*, (Computer Science Press, Rockville, Ma., 1988).

[25] T. M. Murali and T. A. Funkhouser, "Consistent solid and boundary representations from arbitrary polygonal data", in *Proceedings of 1997 Symposium on Interactive 3D Graphics* , (1997) pp. 155-162.

[26] . G. Vanecek, "Set operations on polyhedra using decomposition methods", Ph.D. Thesis, University of Maryland, 1989.

[27] K. Weiler "Polygon comparison using a graph representation", *Computer Graphics*, **13** (August 1980) 10-18.

[28] K. Weiler, "Topological structures for geometric modeling", Ph.D. Thesis, Rensselaer Polytechnic Institute, N. Y., August 1986.

[29] G. Wyvill, C. McPheeters, and B. Wyvill, "Data Structure for Soft Objects", *The Visual Computer,* vol 2, no. 4, pp. 227-234, 1997.

[30] D. Zorin and P. Schröder, editor, *Subdivision for Modeling and Animation,* ACM SIGGRAPH'2000 Course Notes no. 23, July, 2000.