An Interactive Shape Modeling System for Robust Design of Functional 3D Shapes

Ergun Akleman
Texas A&M University, USA

Jianer Chen
Texas A&M University, USA

Vinod Sirinivasan
Texas A&M University, USA

Abstract
In Architecture, it is essential to design functional and topologically complicated 3D shapes (i.e. shapes with many holes, columns and handles). In this paper, we present a robust and interactive system for the design of functional and topologically complicated 3D shapes. Users of our system can easily change topology (i.e. they can create and delete holes and handles, connect and disconnect surfaces). Our system also provide smoothing operations (subdivision schemes) to create smooth surfaces. Moreover, the system provides automatic texture mapping during topology and smoothing operations.

We also present new design approaches with the new modeling system. The new design approaches include blending surfaces, construction of crusts and opening holes on these crusts.

Keywords
Modeling, Shape Design, Sculpting, Computer Aided Geometric Design

1 Introduction

A mathematical class called orientable 2-manifolds is essential to describe functional 3D shapes with arbitrary topological structure (i.e. having any number of holes and columns, handles). 2-manifold shapes are essential for physical simulations. For instance, in order to simulate pouring water from a computer generated teapot shape, artificial teapot must have a functional (not apparent) spout. Similarly, for heat transfer, there should be no crack, 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. Since the current computer graphics and shape modeling practice are almost exclusively based on polygonal meshes, it is important to guarantee the 2-manifold property of meshes.

Since Architecture deals with real shapes, for architectural design it is extremely important to guarantee orientable 2-manifold property of shapes. Unfortunately, current modeling paradigms do not support 2-manifolds. For instance, the most commonly used operations in architectural modeling, set operations can result in non-manifold surfaces. Many existing data structures in mesh modeling are specifically developed in such a way that they can represent non-manifold surfaces resulting from the set operations (Hoffman 1989; Hoffman and Vanecek 1990). Because of this fundamental problem, in the process of obtaining the initial control mesh, unwanted artifacts can be generated. These artifacts include wrongly-oriented polygons, missing polygons, cracks, and T-junctions (Barequet and Kumar 1997; Murali and Funkhouser 1997).

The difficulty of topological modeling of orientable 2-manifold polygonal meshes is a very well-known challenge in shape modeling and computer graphics (Hoffman 1989; Hoffman and Vanecek 1990; Ferguson et al. 1992; Welch and Witkins 1994; Takahashi et al. 1997; Fomenko and Kunii 1997). There has been extensive research on the development of effective structures for representing and modeling orientable 2-manifolds (Baumgart 1975; Guibas and Stolfi 1985; Hoffman 1989; Hoffman and Vanecek 1990). We have recently introduced Doubly Linked Face List (DLFL) (Chen 1997; Akleman and Chen 1999; Akleman et al. 2000) that can provide an effective solution to this challenge. DLFL data structure always corresponds to a valid 2-manifold (i.e. DLFL guarantees that it will never create a non-2-manifold structure). We also introduced edge insertion and edge deletion operations to change the topology of 2-manifold meshes (Akleman and Chen 1999; Akleman et al. 2000).

This paper presents an interactive and robust orientable 2-manifold modeling system based on DLFL data structure. For the development of the system, we have introduced a set of powerful, user-friendly, and effective user-interface level operations. With these user-interface level operations, users of our system can easily create and delete holes and handles, connect and disconnect surfaces. Our system is topologically robust in the sense that users will never create invalid 2-manifold mesh structure with these operations.

We have developed new design approaches for topological modeling. The new design approaches include solid-like modeling, surface blending and crust construction. By using new design approaches we have created various shapes using our system. To interactively model these shapes is extremely difficult without our system. Figures 1, 2 and 3 represent examples of crust modeling.

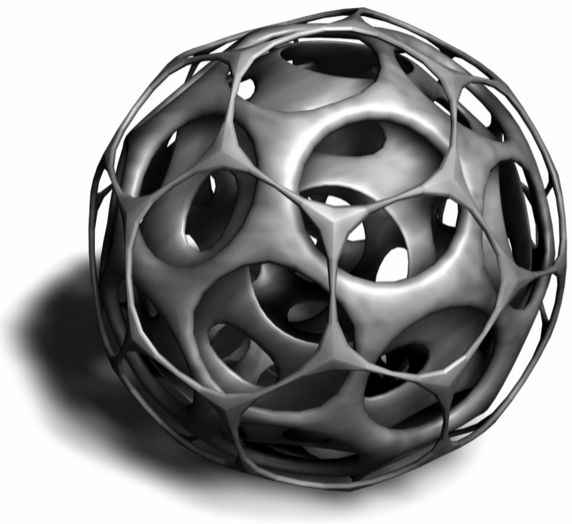

Figure 1. An example of 2-Manifolds with high genus.

The inspiration for the shape 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. As it can be seen from a 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 numbers 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 structure by constructing it from a sphere or from a torus with given texture coordinates. Figure 4 is an example of texture mapping when topology changes. Note that this example also demonstrates surface blending (i.e. our system can be used to create bloby shapes which are similar to implicit bloby shapes (Blinn 1982; Wyvill et al. 1986) with polygonal meshes).

Another benefit of modeling 2-manifolds with topology changes is that it gives the users the feeling of working with solid models. 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.

An additional benefit of our system is that we can smooth out surfaces with subdivision schemes. Recently, some architects such as Frank Gehry started to design buildings that consist of smooth shapes. Architects generally use free form surfaces

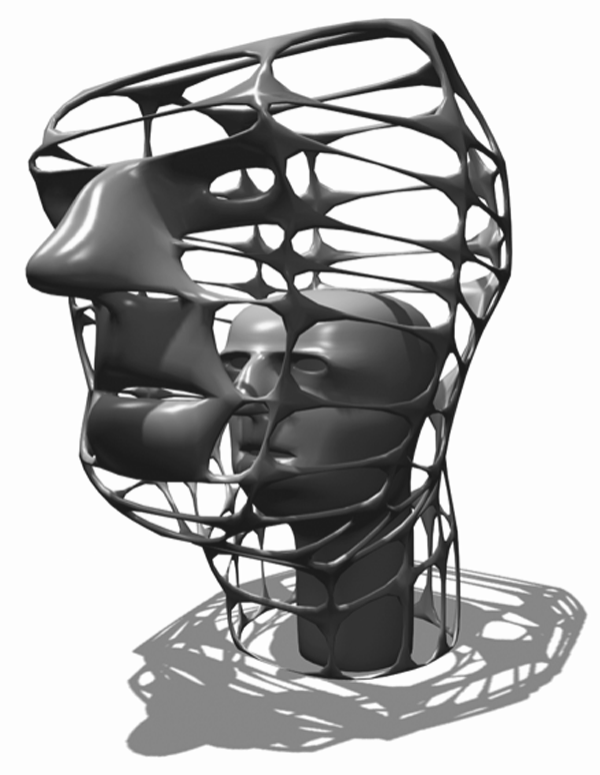

Figure 2. An example of 2-manifold that is designed by our system

Figure 3. A teapot created by using our system.

such as NURBs, B-splines or Bezier to convert their production pipeline from free-forms to subdivision surfaces.

There are two reasons behind the superiority of subdivision surfaces. (1) Subdivision surfaces are a generalization of free-form surfaces (i.e. they can represent any free-form surface). (2) Subdivision schemes can smooth out any 2-manifold (i.e., they can smooth out shapes with any number of holes, columns or handles). On the other hand, free-form surfaces can only provide topologically simpler shapes. With free-form surfaces, to design topologically complicated shapes (shapes that have more than one hole, column or handle) the designers have to be extremely careful not to surfaces. However, for the creation of smooth shapes subdivision schemes are superior to Free-form surfaces (Catmull and Clark 1978; Doo and Sabin 1978; Zorin and Schroder 2000).

Subdivision surfaces are re-introduced to Computer Graphics by a Computer Animation company Pixar (The maker of the computer animated movies such as Toy Story and Toy Story II) in their Academy Award-winning short film "Geri's Game" (Zorin and Schroder 2000). Pixar has already converted most of the production pipeline to subdivision surfaces, and the other computer graphics and special effect companies such as Disney Feature Animation and Industrial Light and Magic (ILM) started create a non-functional surface. This limitation of free-form surfaces restricts their usage in architectural design. We, therefore, expect subdivision surfaces will eventually become popular among the Architects who use smooth shapes in their designs.

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

The paper is organized as follows. In section 2, we introduce user-interface operations for topological modeling and explain how to use them. In section 3, we present new design paradigms for topological modeling such as solid-like modeling, crust modeling and surface blending. In section 4, we describe implementation details of our system. Section 5 concludes the paper.

2 User-Interface Operations for Topological Modeling

We have developed a set of powerful and effective operations at a high level of user-interface. Using these user-interface level operations, users can perform a large set of topological and smoothing operations effectively on a given 2-manifold structure. Moreover, 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 operations implemented in our 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 operation prevents users from unintentionally leaving unwanted vertices in edge deletion operations.

InsertEdge(e) inserts a new edge between two corners (vertex-face pairs). If two corners belong to the same face, InsertEdge(e) subdivides the face. In case the operation InsertEdge is inserting an edge to corners of two different faces, 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, 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. The new pipe either makes a new handle for the 2-manifold (in case the pipe runs from outside of the 2-manifold), or makes a hole in the 2-manifold (in case the pipe runs from inside of the 2-manifold). The geometric effect of this operation is that a very small new pipe is run between the two faces, which merges the two faces into one (Akleman et al. 2000). 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 operations that produce high quality handles and holes, we developed a *CreateHandleHole* operation at the user-interface level. When two different faces are specified for running a pipe between them, the CreateHandleHole operation runs a much thicker pipe between the two faces, as shown in Figure 5. Implementation of this operation based on the fundamental operations is also straightforward. Instead of a single call to the fundamental operation 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 operation CreateHandleHole is that the size of the handle or the hole can actually be controlled; it is simply decided by the corresponding face sizes.

We also found it useful to provide a *SubdivideEdge* operation to users. This operation simply subdivides an edge into two edges by creating a new valence-2 vertex at the middle of the edge. This operation allows users to create new vertices in a 2-manifold without changing its topological structure, which is heavily used in the implementation of various subdivision schemes.

Additional operations, such as *CreatePolyhedra* and *Subdivision*, are also developed and implemented. Using the CreatePolyhedra operations, users are able to create regular polyhedra, such as cubes, tetrahedra, octahedra, icosahedra, dodecahedra and polyhedral torii. Using the Subdivision operation, users can apply a variety of subdivision schemes to a 2-manifold, including a number of vertex-insertion schemes, such as Catmull-Clark scheme (Catmull and Clark 1978) and, a number of corner-cutting schemes, such as Doo-Sabin scheme (Doo and Sabin 1978). The implementations of these subdivision schemes are based on the previously developed algorithms of the schemes in the original DLFL structure (Akleman et al. 2000; Akleman et al. 2001).

Subdivision operation is particularly important in Architectural design to model smooth shapes. Since subdivisions are a generalization of NURBs, B-splines and Bezier surfaces and can smooth out any 2-manifold (Catmull and Clark 1978, Doo and Sabin 1978, Zorin and Schroder 2000), we expect that architects who want to design smooth shapes will eventually start to use subdivisions instead of free-form surfaces. As we have mentioned earlier, a similar conversion are already occurring right now in computer animation and special effect companies.

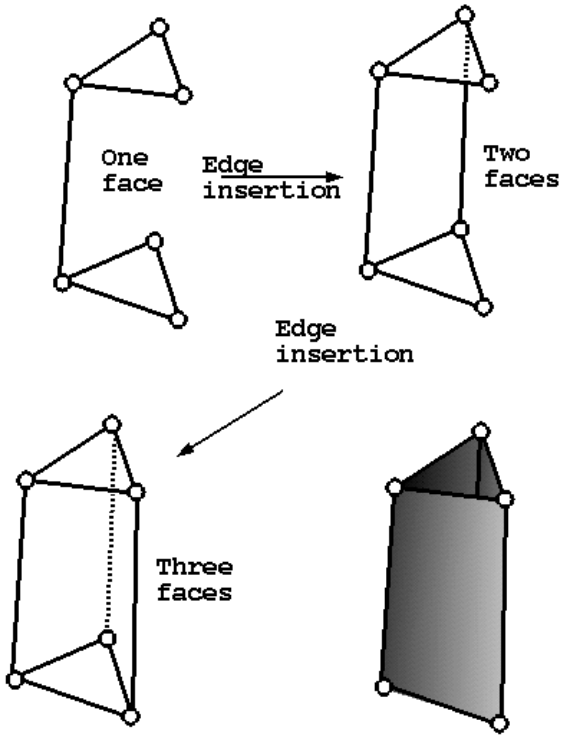

Figure 5. Creation of a high quality handle/hole by inserting a set of edges.

3 New Design Methods for Topological Modeling

Although it is a polygonal surface modeler, with our system users can create shapes that look like being created by a different type of modeler, such as solid or implicit modelers. This section introduces new design methods that are closely related to our system. We present a number of examples to illustrate these methods.

3.1 Solid-Like Modeling and Non-Manifold Looking 2-Manifolds

One of the unexpected results from user-interaction with our system is that users may feel that they are dealing with solid shapes. We show why this impression is formed by the examples given in Figures 6 and 7.

In the first example, we start with a surface that looks like a wall made up of 9 cubes (see Figure 6(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.

Figure 7 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 CreateHandleHole operation. 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 7(E)), the resulting shape looks like a non-manifold. This is 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 7(E)), which is only applicable to 2-manifold struc-

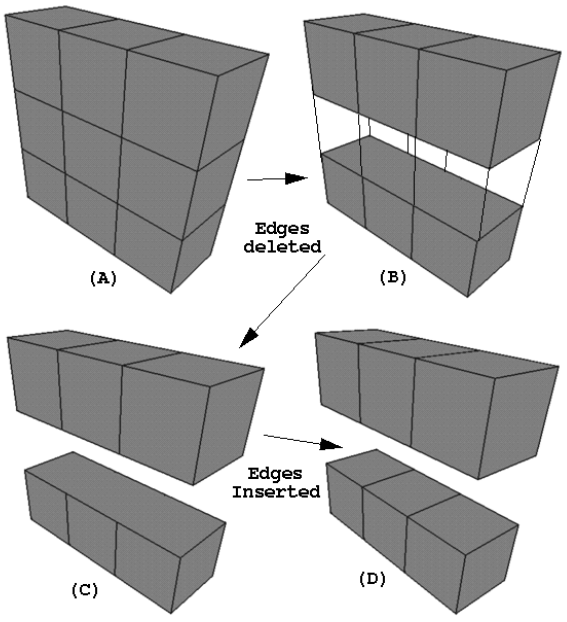

Figure 6. Dividing a simple 2-manifold surface that looks like a wall made up of 9 cubes into two separate surfaces.

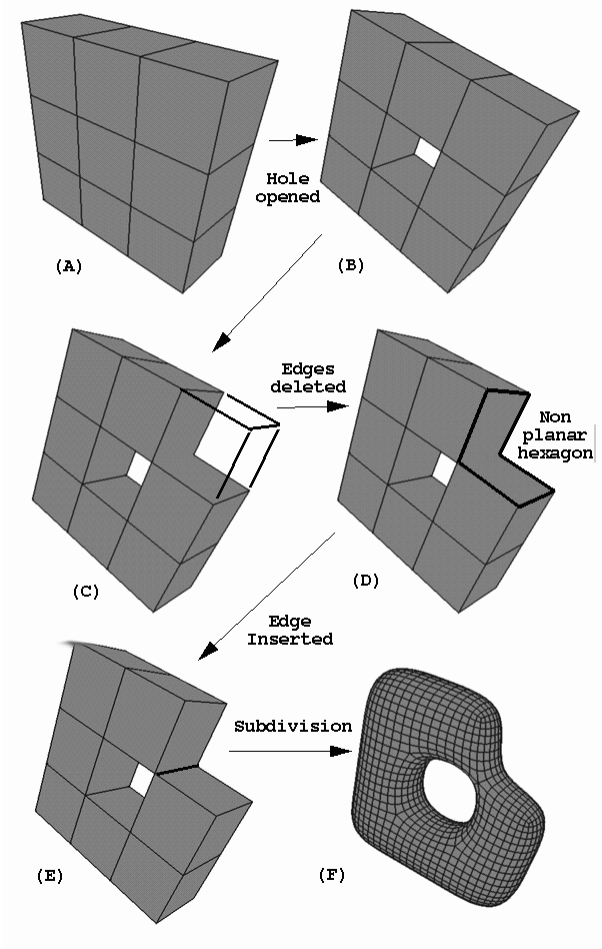

Figure 7. Modeling a simple 2-manifold that looks like a non-manifold.

tures. 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.

3.2 Surface Blending (Implicit-Like Modeling)

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 CreateHandleHole operation, 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 (One example of truncated icosahedron is shown in Figure 9(A). This polyhedron consists of 12 pentagons and 20 hexagons similar to soccerballs and it is the basic structure behind the geodesic dome.) and connect the two nearest pentagons by the CreateHandleHole operation. Then, we apply Catmull-Clark subdivision (Catmull and Clark 1978) to both initial two truncated icosahedra and connected surface to get two images in Figure 4.

Figure 8 shows another example of implicit-like blending: tree-like branching shapes (Bloomenthal 1985). 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).

3.3 Crusts Modeling with Holes

By using our system it is easy to create shells that are similar to a coconut crust and open holes on this crust. 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 further move the positions of some vertices). This operation creates two nested surfaces. (2) Reverse the normals of the smaller second mesh. This operation 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 CreateHandleHole operation. This operation connects two surfaces and makes them a single surface. Then, additional holes can be created using the CreateHandleHole operation.

Figure 9 demonstrates two different models created by crust 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 CreateHandleHole operation 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 (Akleman et al. 2001) 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

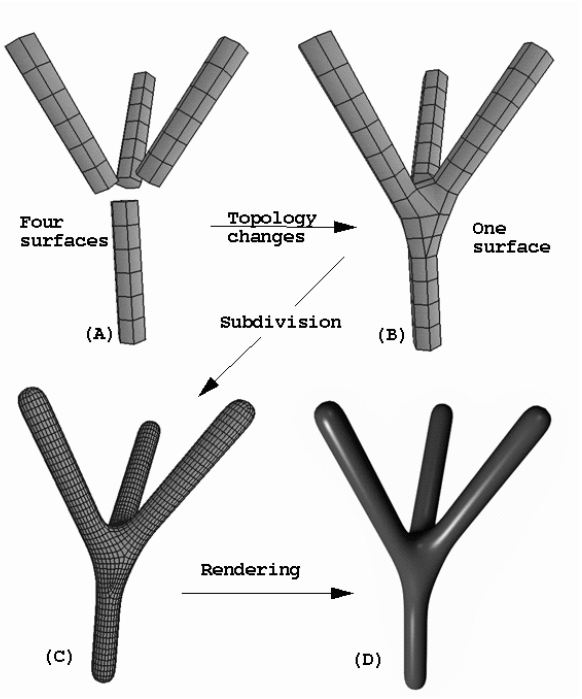

Figure 8. Implicit-like modeling.

(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 crusts. 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.

3.4 Automatic Texture Mapping During Topology Changes

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 (Catmull and Clark 1978) is applied to texture coordinates, textures appear to move. Therefore, instead of using Catmull-Clark, we have used an interpolation scheme for texture coordinates.

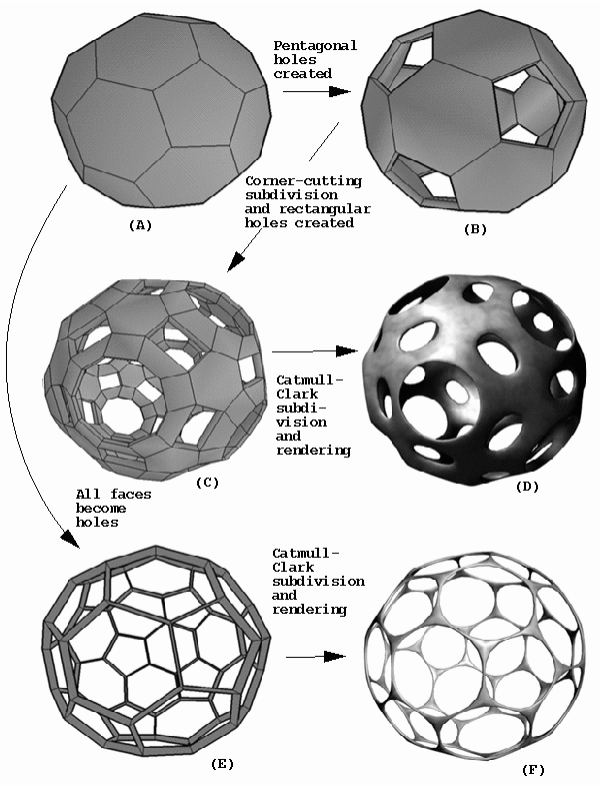

Figure 9. Example of modeling a crust.

4 Implementation Details

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 Gooch et al.'s non-photorealistic lighting model for rendering since it does not leave any part of the object completely in shadow (Gooch et al. 1998). All the images that show wireframe, such as the ones in Figure 7, 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.

5 Conclusion and Future Work

In this paper, we have developed a system for topological modeling. A set of powerful, user-friendly, and effective operations has also been developed at the level of user-interface. Users of our system can perform topology changes and smoothing with these user-interface level operations. Moreover, because our system is based on a 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 operations.

We have also introduced a number of design methods, e.g., surface blending or crust modeling, to create shapes with the new modeler. By using these methods, we have interactively 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.

Our system can be extremely useful for architectural design since in Architecture it is required to design functional shapes with complicated topology. In the future, we will continue to improve the interface of our system by introducing new design methods and developing high-level operations for these methods.

References

E. Akleman, J. Chen. (1999). "Guaranteeing the 2-Manifold Property for meshes with Doubly Linked Face List", International Journal of Shape Modeling, Volume 5, No 2, pp. 149-177.

E. Akleman, J. Chen and V. Srinivasan. (2000). "A New Paradigm for Changing Topology During Subdivision Modeling," Pacific Graphics 2000.

E. Akleman, J. Chen, F. Eryoldas and V. Srinivasan. (2001). "Handle and Hole Improvement by Using New Corner Cutting Subdivision Scheme with Tension," Proceedings of Shape Modeling 2001.

J. Chen, "Algorithmic Graph Embeddings". (1997). Theoretical Computer Science, no. 181, 1997, pp. 247-266.

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

R. H. Bartels, J. C. Beatty, and B. A. Barsky. (1987). An Introduction to Splines for Use in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers, Los Altos, CA.

B. J. Baumgart. (1972). "Winged-edge polyhedron representation", Technical Report CS-320, Stanford University.

B. J. Baumgart. (1975). "A polyhedron representation for computer vision", in 44th AFIPS National Computer Conference, pp. 589-596.

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

J. Bloomenthal. (1985). "Modeling the Mighty Apple", Computer Graphics, vol 19, no. 3, pp. 305-311.

E. Catmull and J. Clark. (1978). "Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes", Computer Aided Design, no.10, pp. 350-355.

D. Doo and M. Sabin. (1978). "Behavior of Recursive Subdivision Surfaces Near Extraordinary Points", Computer Aided Design, no. 10, pp. 356-360.

H. Ferguson, A. Rockwood and J. Cox. (1992) "Topological Design of Sculptured Surfaces", Computer Graphics, no. 26, pp.149-156.

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

L. Guibas, J. Stolfi. (1985). "Primitives for the manipulation of general subdivisions and computation of Voronoi diagrams", ACM Transaction on Graphics, no. 4, pp. 74-123.

C. M. Hoffmann. (1989). Geometric & Solid Modeling, An Introduction, Morgan Kaufman Publishers, Inc., San Mateo, CA.

C. M. Hoffmann and G. Vanecek. (1990). "Fundamental techniques for geometric and solid modeling", Manufacturing and Automation Systems: Techniques and Technologies, no. 48, pp. 347-356.

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

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

B. T. Stander and J. C. Hart. (1997). "Guaranteeing the Topology of an Implicit Surface Polygonization for Interactive Modeling", Computer Graphics, no. 31, pp. 279-286.

S. Takahashi, Y. Shinagawa and T. L. Kunii. (1997). "A Feature-Based Approach for Smooth Surfaces", in Proceedings of Fourth Symposium on Solid Modeling, pp. 97-110.

D. Zorin and P. Schroder, co-editors. (2000). Subdivision for Modeling and Animation, ACM SIGGRAPH'2000 Course Notes no. 23.

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

W. Welch and A. Witkin. (1994). "Free-Form Shape Design Using Triangulated Surfaces", Computer Graphics, no. 28, pp. 247-256.