Interactive Rind Modeling

ERGUN AKLEMAN *
Visualization Laboratory
College of Architecture

VINOD SRINIVASAN
Visualization Laboratory
College of Architecture

JIANER CHEN †
Department of Computer Science
College of Engineering

Texas A&M University

Abstract

In this paper, we describe a technique, with roots in topological graph theory, that we call rind modeling. It provides for the easy creation of surfaces resembling peeled and punctured rinds. We show how the method's two main steps of 1) creation of a shell or crust like the rind of an orange, and 2) opening holes in the crust by punching or peeling can be encapsulated into a real time semi-automatic interactive algorithm. We include a number of worked examples, some by students in a first modeling course, that demonstrate the ease with which a large variety of intricate rind shapes can be created.

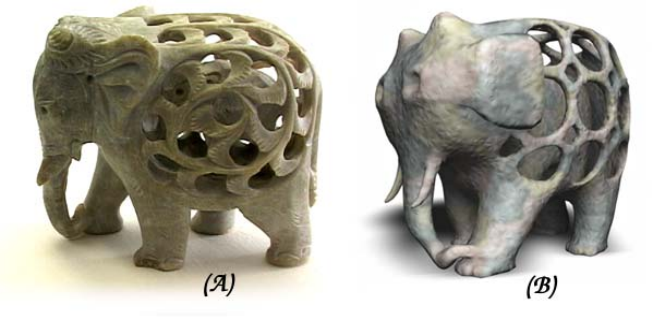


Figure 1. An example of nested sculptures from India. (A) shows a real nested elephant sculpture. (B) is a rind model created using our system and rendered with bump mapped textures.

1 Motivation

The inspiration for this paper came from Escher's drawings of rind shapes [16], and the intricate nested carved sculptures of the Far East. Figure 1A is an example of a nested elephant sculpture from India. A second elephant can be seen inside the first through the holes on outer surface. We have also seen nested balls from China that consist of up to 16 rotateable balls carved inside of each other.

Although our inspiration came from art, such rind shapes are extremely useful in industrial applications. Many man-made objects are rind shaped. Examples are endless and include bottles, teapots, masks, boxes and even houses. Also, rind shaped surface meshes can be functional models and therefore can be used in physical simulations.

We have developed a user friendly manifold modeling method that we call *rind modeling*. It allows us to construct easily very high genus rind shaped manifold surfaces. For example, the outer shape in Figure 1B is a rind model created using our method and rendered with bump mapped textures. Figure 2 shows examples of spherical rind shapes. As seen in the figure, shapes similar to nested chinese ball sculptures can be obtained by scaling and rotating the spherical rind shapes. The shapes in Figure 2(B) and (C) are motivated by Escher's rind objects [16].

Rind modeling consists of two steps: in the first (automatic) step, for any given 2-manifold mesh surface an offset surface is created based on a user defined thickness parameter. As a result of this step, from one surface two similar surfaces, which can be considered as a shell or a crust are created. The second (interactive) step consists of two modes, hole punching and peeling, similar to punching holes in a coconut husk or peeling an orange rind. Holes are punched by single mouse clicks.

The rind modeling method can also be used to create shapes that do not necessarily look like rind shapes. An example of such uses is shown in Figure 3. The surface in this figure looks more like an extruded surface than a rind surface, but it cannot be created as a simple extrusion because of its branching structure.

*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.

†Address: Department of Computer Science, College Station, TX 77843-3112. email: chen@cs.tamu.edu. Supported in part by the National Science Foundation under Grant CCR-9613805.

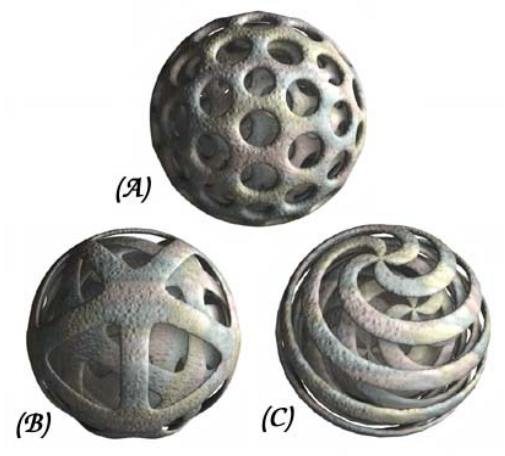


Figure 2. Three examples of spherical rind shapes. The nested structures are obtained by scaling and rotating the outer rind shapes.

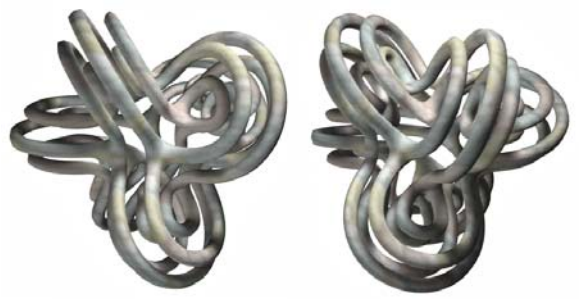


Figure 3. Two views of a shape that can be created by our method, but does not look like a rind shape.

The teapot shown in Figure 4 is an example of a functional model that can be created by our method. As can be seen from the x-ray and cut away images, this teapot has a *real* not just an apparent hole to let the water pour from the spout. Because of the hole in the spout, this teapot could be used in physical simulations. Moreover, as can also be seen in the images, there exists an additional hole inside of the handle of the teapot, i.e., this model has a rind shape.

2 Previous Work

Creation of very high genus manifold surfaces has always been a research interest in computer graphics and shape modeling. Ferguson, Rockwood and Cox used Bezier patches to create high genus 2-manifold surfaces [17]. Welch and Witkins used handles to design triangulated free-form surfaces [31]. Using Morse operators and Reeb graphs, Takahashi, Shinagawa and Kunii developed a feature-based approach to create smooth 2-manifold surfaces with high genus [29].

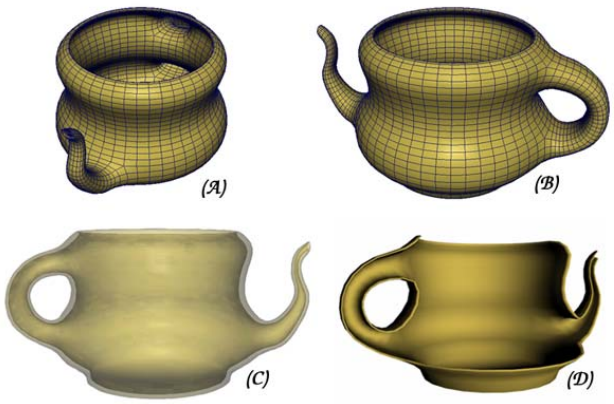


Figure 4. A rind-shaped teapot created using our system. (A) and (B) show two different views of the manifold mesh. (C) is an x-ray image using transparency. (D) is sliced to show the interior. (This slice was created using our system in peeling mode.)

We recently introduced Doubly Linked face List (DLFL) data structure and a topologically robust mesh modeling approach [1, 2, 12][1] based on topological graph theory [13, 23]. Our EDGEINSERT and EDGEDELETE [2] operators can effectively change the topology of a manifold mesh by inserting and deleting handles. They can be used to effectively implement subdivision schemes and allow topology change during subdivision modeling [3, 4, 5]. We have recently developed a user interface [6, 7] and theoretically shown [8, 12] that all and only orientable 2-manifold structures can be created using two simplest Euler operations MVFS (make a surface with a single vertex and a single face) and KVFS (inverse of MVFS) [27] along with EDGEINSERT and EDGEDELETE. Moreover, these four operators can be efficiently implemented [12] on almost every mesh data structure including winged-edge, [9], half-edge [27] and quad-edge[24].

These results suggest that software development for mesh modeling with a topologically guaranteed orientable manifold property can be greatly simplified and high level and intuitive topology change operators can be constructed using only these four operators. A recently introduced high-level operator called CREATEPIPE [7] is an example of such

[1] For detailed discussions and theoretical comparisons with Solid Modeling, Euler operators, and existing data structures see [2] and [12]. These papers are also available at www-viz.tamu.edu/faculty/ergun/topology.

intuitive topology change operators. Let $\mathcal{M}$ denote a 2-manifold mesh and let v, f and e denote a vertex, a face and an edge respectively. We define a *corner* of the mesh to be a vertex-face pair, $c = \{v, f\}$ if f contains the vertex v. The CREATEPIPE(c_1, c_2) operator connects two faces which contain the corners c_1 and c_2 such that there is an edge between c_1 and c_2 and there is an edge between other matching corners (in traversal order starting from c_1 and c_2) of the two faces, as shown in Figure 5. In effect this operation automatically creates a pipe made up of only one segment whose ends are defined by the original faces. This automatic process helps the users to intuitively understand the topology change as creating a pipe between two faces.

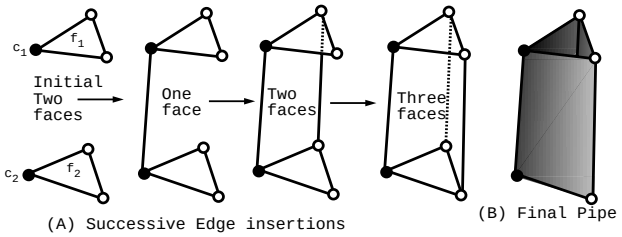


Figure 5. An example of creation of a pipe (a high quality handle/hole) by inserting a set of edges. If this pipe goes through the inside of the manifold surface (as happens in our application in this paper), it punches a hole.

The CREATEPIPE operator is extremely useful for developing mesh design methods to create very high genus manifold surfaces. For instance, by combining the CREATEPIPE operator with subdivision schemes, shapes that resemble implicit surfaces can be created [8, 7]. The CREATEPIPE operator can also be extended to construct multi-segment, curved handles [30]. In this paper, we use the CREATEPIPE operator to construct rind shaped high-genus manifold surfaces. We have first created rind shaped manifold meshes using a naive algorithm [7], as an example see the cover image of Proceedings of International Conference on Shape Modeling and Applications 2002, *SMI'02*.

3 Problem Description

It is straightforward to develop a naive algorithm for the construction of high-genus rind surfaces based on the CREATEPIPE operator. This algorithm would proceed in two stages.

Stage 1: Offset surface creation. This stage consists of two steps.

1. Duplicate the initial mesh and move the vertices of the new mesh so that they will lie completely inside the first mesh (if the object is not convex or star-like, it may be necessary to further move some vertices). This operation creates two nested surfaces, which consist of the *initial mesh* and the newly created *offset mesh*.

2. Reverse the normals of the faces of the offset mesh. This operation changes the inside and outside of a 2-manifold mesh by changing the rotation orders of faces.

Using only this stage, it is possible to create rind models. Although such a model would be valid, in the sense that both outside (initial) and inside (offset) surfaces are manifold, it will not be interesting without punching holes or peeling away parts of the rind. Without those holes, it is not possible to see inside of the shapes or create nested structures such as those shown in Figures 1 and 2.

The second stage of the naive algorithm is needed to create such high genus shapes. In this stage, the users can choose to either simulate punching holes or peeling away parts of the rind.

Stage 2A: Hole punching. This stage consists of two topologically different steps. However, users perceive both steps as being same, namely, punching holes.

1. Connect two corresponding faces using the CREATEPIPE operation. This operation connects initial and offset surfaces and makes them a single surface. Formally, this first CREATEPIPE operation connects the two surfaces (i.e. changes the topology) but does not increase the genus [7, 12].

2. Using the CREATEPIPE operation, create additional holes. The CREATEPIPE operation after the first one both changes the topology and increases the genus by one [7, 12].

Stage 2B: Peeling. Punching holes does not automatically create a peeled effect. Punching holes in two neighboring faces leaves an infinitely thin wall or slab (like a thin membrane) between two neighboring holes. Users have to delete such infinitely thin walls to create a peeled look as shown in Figure 6.

An example of shapes we have created with the naive algorithm using hole punching from our earlier paper is shown in Figure 7. One might think that almost any very high-genus manifold rind shape can be created by a skilled user with this naive algorithm. This observation is theoretically correct, but, practically, because of several usability problems it is not easy to create complicated rind surfaces even by skilled users. The usability problems we have identified can be summarized as follows.

1. *Usability problem with offset surface creation.*

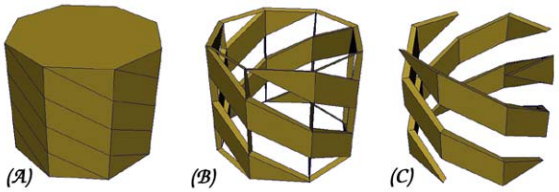


Figure 6. An example of the need for peeling. (A) shows an initial mesh and (B) shows infinitely thin walls left over by hole punching. The peeling effect shown in (C) is obtained only after deleting these infinitely thin walls.

Figure 7. A rind teapot model that is constructed with the naive algorithm. (A) is actual rendering with bump mapped textures and (B) is an x-ray image.

It is not easy to create an offset surface by hand for complicated initial meshes. Although the model that is shown Figure 7 is valid, in the sense that the surface is manifold and the teapot could hold water, the model is disappointing because the thickness of the rind is highly nonuniform.

To create a uniform rind thickness by hand is almost impossible for high-genus initial meshes. For instance, in the teapot shapes in Figure 7 the initial mesh already had a handle. It is very difficult to create an offset surface that gives an additional hole inside this handle of the teapot.

2. *Usability problem with hole punching.*

   It is not easy to select two corners of two related faces in both the initial and offset surfaces for opening holes with the CREATEPIPE operator. It is especially difficult to select a correct corner in the offset surface since this surface is inside and partially visible only through holes. This difficulty is even more pronounced in the application of the first CREATEPIPE operation that connects two surfaces since initially the offset surface is completely invisible. (One reviewer pointed that this problem is not essential since it could simply be solved using transparency to see the interior of a rind model.)

3. *Usability problem with peeling.*

   The peeling stage of the naive algorithm requires users eliminates the walls . Each of these walls consists of two quadrilaterals like a folded paper. Since all edges are straight, this wall looks like a two sided quadrilateral but it is not (note that we only deal with orientable 2-manifolds without a boundary. A two sided quadrilateral is not a legal face and cannot be created use of the four fundamental operations.)

   Having said that, now, let us assume that two edges of one of the quadrilaterals are curved as shown in Figure 8A. Although this figure geometrically does not make sense, it helps to conceptually visualize the structure of the infinitely thin wall. Note that this infinitely thin wall is in fact a handle and this handle can be deleted by deleting two edges e_1 and e_2 as shown in Figure 8. Deletion of e_1 combines the two quadrilaterals and creates a hexagonal face [7]. As we have mentioned earlier, this hexagon is still a handle. Only after the deletion of e_2, separating this hexagon into two two-gons, is the handle eliminated.

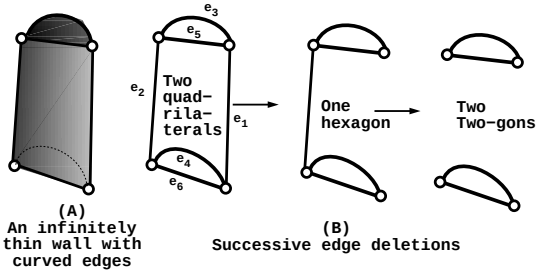


Figure 8. Deletion of a an infinitely thin pipe by deleting only two edges. Initially this infinitely thin pipe consists of two quadrilaterals.

Note that this deletion process is hard for the users. The requirement to delete the first two edges e_1 and e_2 shown in Figure 8 can be done during an interactive session since the pipe deletion is a visual change. Moreover, deletion of the first of such edges, e_1, can be easily performed. However, after the first edge deletion, as shown in Figure 8, a strange hexagon which cannot be adequately rendered is created. Because of this rendering problem, it can be difficult to delete the second edge which becomes visible only from some viewpoints.

The resulting two two-gons, which are created after e_1 and e_2 are deleted, do not carry any extra visual (or geometric) information (they look exactly like straight

edges). If these left-over two-gons are not deleted subdivision schemes would not be able to adequately smooth out the meshes. Therefore, it is important to delete these left-over two-gons. Figure 9 illustrates how deletion of one edge of a two-gon eliminates it. In the case of Figure 8 deleting e_3 and e_4 will eliminate two-gons. However, deleting e_3 and e_4 is not easy. Two-gons are not visible as they look exactly like straight edges. It is not easy to delete something that could not be visible or detected. Moreover, the edges of two-gons are usually very short and therefore it is very difficult to select and delete these edges by hand. In other words, even if the users try to delete all such edges, some may still be overlooked. As we mentioned earlier, such left-over edges later create a problem when a subdivision scheme is applied as shown in Figure 10. As a result, it is better to delete pipes automatically, preferably with a very simple and fast operation.

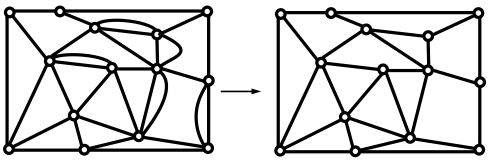


Figure 9. Two-gons sre removed by deleting one of their edges. In order to show the actual mesh structure, one of the edges of every two-gon is drawn curved.

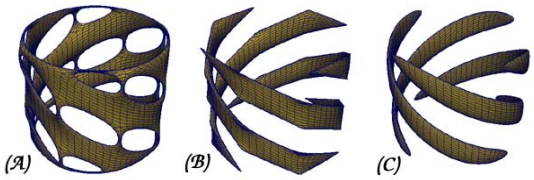


Figure 10. The effect of two-gons in smoothing with subdivision. (A) shows a subdivided version of 6B and (B) and (C) show subdivided versions of 6C with and without two-gons. Note that the version with two-gons shows tangent discontinuities. In all cases, we applied Catmull-Clark [11] subdivision twice.

The approach we present below solves all of these problems with a very simple real time interactive method.

4 Methodology

We solve the usability problems with a semi-automatic approach. When the user selects a manifold surface and then chooses rind modeling mode, the system automatically creates an offset surface based on the value of a user set thickness parameter. Then users either punch holes or peel by simply selecting faces. While peeling, the infinitely thin pipes between neighboring holes are automatically eliminated.

The following is our semi-automatic method:

Automatic step: Offset surface creation. This is the first operation after the user selects rind modeling mode.

1. Duplicate the initial mesh to create an offset mesh, reverse the normals and compute average unit normals for each vertex.

2. Create a *correspondence table* that records the corresponding faces of the initial and offset meshes. The correspondence table also includes one corresponding corner for each face.

3. Move each vertex of the offset mesh in the direction of its average normal vector a distance equal to the thickness value.

   Remark. The offset surface construction problem for 2-manifold meshes is very different from offset surfaces in solid modeling [21, 20] since the shapes of faces of 2-manifold meshes do not have to be well-defined. For 2-manifold meshes we cannot describe a distance function and, therefore, we cannot really define a theoretically "correct" offset surface. Thus, any procedure to create offset surfaces for 2-manifold meshes must be somewhat ad hoc. This procedure is practically useful and succesfully creates a practically acceptable offset surface for small enough values of thickness parameter.

   The procedure may result in self intersection for high values of the thickness parameter. It is possible to accomodate the self intersection by changing topology and mesh structure as shown in Figure 11. However, for hole punching we want both the initial and offset surfaces to have exactly the same mesh structure. Thus, we avoid self intersections by simply using smaller thickness values. Of course, another advantage of this procedure is that it is extremely simple, and therefore, suitable for real time applications.

After the offset surface has been (automatically) created, the interactive step is initiated. During the interactive step, the system can be in two modes: hole punching and peeling. The user can change the mode during the interactive

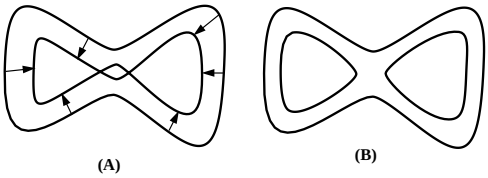


Figure 11. Avoiding the self intersection that is shown in (A) requires a change in topology as shown in (B).

process. As soon as the offset surface is created users see a message that says "select a face to punch holes" or "select a face to peel" depending on the chosen mode. Then they can either punch holes (or peel depending on mode) by simply selecting faces of either the initial or offset surfaces.

Interactive step: Hole punching mode. The algorithm for punching one hole is extremely simple.

1. Find the selected face in the correspondence table.
2. Using the two faces and two corners obtained from the table, apply CREATEPIPE.

   Remark. If the users select a face that is not in the correspondence table, the procedure ignores the selection. Note that each hole creation operation deletes two old faces and creates k new faces (quadrilaterals) where k is the number of sides of the selected face. These newly created quadrilaterals must be ignored, but, since these new faces will not be in the correspondence table, this is easy.

Interactive step: Peeling mode. In peeling mode, every infinitely thin pipe created as a result of punching holes is deleted. The procedure for cleaning such unwanted pipes also turned out to be extremely simple based on topological graph theory.

1. Find the selected face in the correspondence table.
2. Using the two faces and two corners obtained from the table, apply CREATEPIPE.
3. For every corresponding two edges of the old (recently deleted) faces of the initial and offset surfaces, check adjacent polygons. If the adjacent polygons are the same, delete all the edges of the quadrilateral face that includes the two corresponding edges.

5 Implementation and Testing

The procedure above is included in our existing 2-manifold mesh modeling system [7] as an option. Our system is implemented in C++ and FLTK [18]. All of our interactive examples were run on an SGI-O2. All the examples in this paper were created interactively using this prototype system. Most images that show wireframe are screen snapshots from our system.

The usability of the system was tested in a graduate level shape modeling course in which a majority of the students had an architecture undergraduate background. Some examples of these students work is shown in Figure 12. Although none of the students had any previous idea about topology, they easily learned to use the software and created a wide variety of high genus manifold meshes as one of their weekly class projects. Figure 12A shows a student's work also motivated by the elephant sculpture shown in Figure 1A. Figures 12B and C show shapes that are created by a student motivated by the shapes of seashells and chinese nested balls. Figure 12D shows two of Escher's rind shaped objects [16] constructed by two other students. The rind shape on the right in Figure 12D is used to create image in Figure 2C.

6 Remeshing

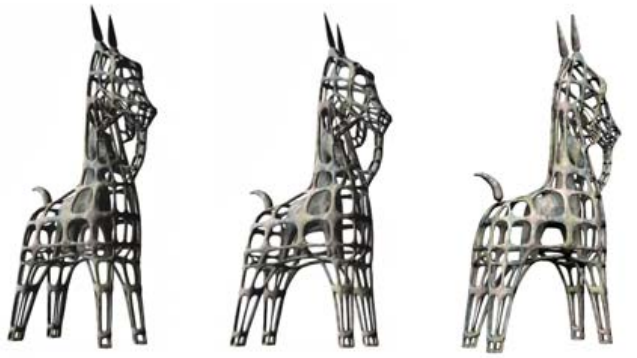
To create interesting looking rind shapes, the mesh structure of the initial manifold surface is very important. Escher's rind shaped objects shown in Figure 12D motivated us to develop remeshing strategies that can create interesting mesh structures.

One of the most useful remeshing strategies turned out to be corner-cutting subdivision schemes [32, 14]. In corner cutting schemes the faces in the first mesh become smaller and connected by quadrilateral paths. Escher used this property to conceptualize the shape on the left in Figure 12D. In terms of rind modeling, Escher's algorithm to create this shape could be formulated (or generalized) as follows:

1. Apply the Doo-Sabin subdivision scheme [14] a few times to a convex polyhedral shape (Escher conceptually started from an Archimedean truncated cuboctahedron.)
2. (Optional) Spheralize the shape (move the vertices of a mesh to a sphere. Since Escher drew a spherical shape, after the application of Doo-Sabin algorithm, there is a need for this spheralization step.)
3. Peel everything except the initial faces and the quadrilateral paths that connect them.

Figure 13 shows a rind shape created starting from a dodecahedron using Escher's algorithm. Ignoring the optional spheralization step in this procedure, complicated rind shapes such as the one in Figure 14 can be created. The shape in Figure 14C is the same as the one in Figure 3.

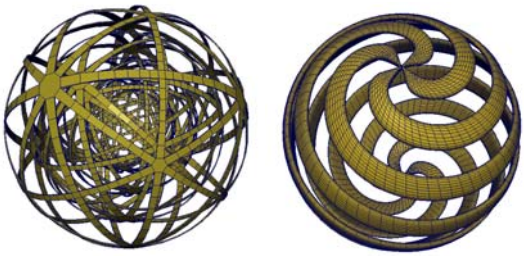
Spiral shaped peeling such as shown in Figures 6 and 12D requires a mesh structure that can be obtained by



A. Horse

B. Seashells

C. Balls



D. An homage to Escher: rind shaped objects [16].

Figure 12. Student work

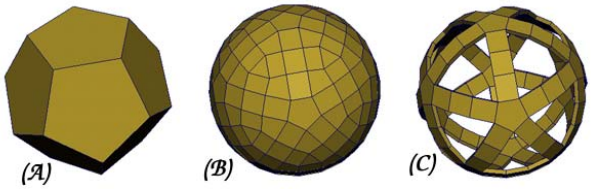


Figure 13. An example of a rind shape created using Escher's algorithm. The initial manifold mesh for rind modeling shown in (B) is obtained by applying Doo-Sabin [14] and spheralization schemes to the dodecahedron shown in (A). We created the shape shown in (C) by peeling everything except the initial faces and the quadrilateral paths that connect them. The subdivided version of (C) is shown in Figure 2B.

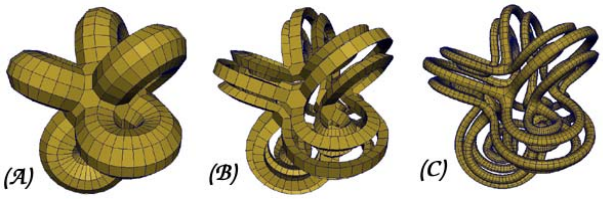


Figure 14. Another example of rind shape created using Escher's idea. The initial manifold mesh for rind modeling shown in (A) is obtained by applying the Doo-Sabin scheme [14] a genus 6 surface. We created the shape shown in (B) by peeling. The shape in (C) is obtained by applying Catmull-Clark [11] to the shape shown in (B). The mesh in (C) is the same as the one that is shown in Figure 3.

remeshing a regular four-connected quadrilateral mesh as shown in Figure 15. Figure 16 shows a spiral peeled rind shape created from a torus.

We have also developed and implemented two new schemes which we call "Honeycomb 1 and 2", that can create useful mesh structures for rind modeling. The new schemes are illustrated in Figures 17 and 19. In both schemes, the algorithms to compute the positions of new vertices give C^1 continuity and allows for shape control with a tension parameter. Honeycomb 1 subdivision scheme is useful for creating flower patterns. This scheme creates flower patterns around extraordinary vertices (which include vertices higher than valence three and faces higher than six sides). Examples of rind shapes that are created by using this property are shown in Figure 18. The honeycomb 2 scheme is useful for creating frames around each face of a mesh structure. Examples of rind shapes that are created by using this property are shown in Figure 20.

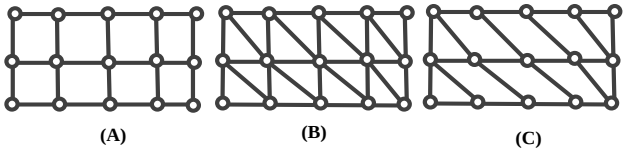


Figure 15. An example of the creation of a mesh structure for spiral shaped peeling from a regular four-connected quadrilateral mesh. Either of the remeshes shown in (B) or (C) can be used.

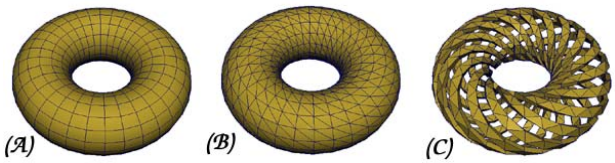


Figure 16. An example of spiral peeling after remeshing the torus shown in (A). (B) is the initial mesh to create the spiral rind shapes shown in (C).

7 Conclusion and Future Work

In this paper, we presented a user friendly manifold modeling method to easily construct very high genus rind shapes. Our semi-automatic method is simple and easy to implement. We have also introduced various remeshing strategies to create initial surfaces that allow the creation of interesting rind surfaces. Development of new remeshing strategies that allow various types of artistic applications might be an interesting future research direction.

Our semi-automatic method simplifies the modeling process, but there are many applications that can be done with complete user control but not with our approach. For instance, users can easily open holes using offset surfaces that do not have the same mesh structure as the initial surfaces as shown in Figure 21. The creation of offset surfaces that do not have the same mesh structure as the initial surfaces is just one part of the problem. It is also necessary to identify corresponding faces for the initial and offset surfaces. This identification can be especially interesting since it may sometimes require one-to-many correspondence between each face of the offset surface and more than one face of the initial surface. If there exists one-to-many correspondence, another interesting problem would be to create a hole from one face to multiple faces.

References

[1] E. Akleman, J. Chen, "Guaranteeing 2-Manifold Property for Meshes", *Proceedings International Conference on Shape Modeling and Applications 1999*, pp. 18-25, Aizu, Japan, March 1999.

[2] E. Akleman and J. Chen, Guaranteeing the 2-manifold property for meshes with doubly linked face list, (Extended version of [1] selected for special issue) *International Journal of Shape Modeling*, Volume 5, No. 2 pp. 149-177, 1999.

[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, HongKong, China. Oct. 2000.

[4] E. Akleman, J. Chen, F. Eryoldas and V. Srinivasan, Handle and hole improvement by using new corner cutting subdivision scheme with tension, *Proceedings International Conference on Shape Modeling and Applications 2001*, (SMI'01), (2001), pp. 32-41, Genova, Italy, May 2001.

[5] E. Akleman, J. Chen, V. Srinivasan and F. Eryoldas, "A New Corner Cutting Scheme with Tension and Handle-Face Reconstruction", (Extended version of [4] selected for special issue) *International Journal of Shape Modeling*, Volume 7, No. 2, pp. 111-121, 2001.

[6] E. Akleman, J. Chen and V. Srinivasan, "An Interactive System for Robust Topological Modeling of Meshes", *Visual Proceedings of ACM SIGGRAPH'2001*, Los Angeles, California, August 2001.

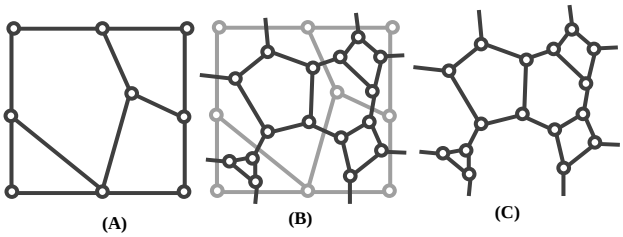


Figure 17. Illustration of honeycomb 1 remeshing algorithm.

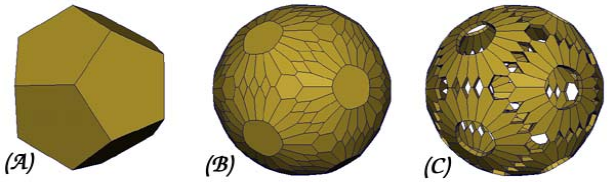


Figure 18. An example of flower pattern creation with honecomb 1 algorithm. To create this pattern, each edge of the dodecahedron shown in (A) is subdivided twice. The flower patterns are obtained by applying honeycomb 1 algorithm twice to the resulting mesh.

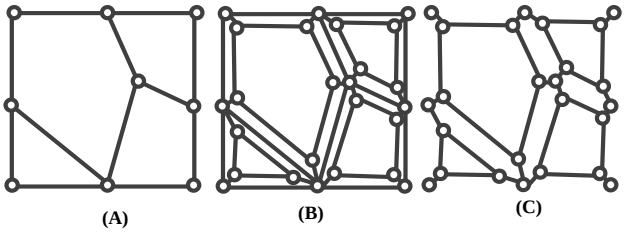


Figure 19. Illustration of honeycomb 2 algorithm.

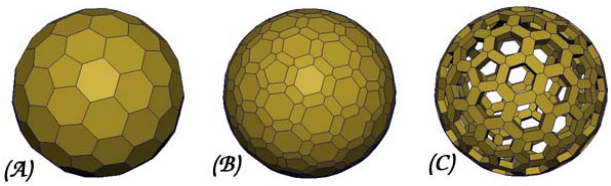


Figure 20. An example of frame creation with honeycomb 2 algorithm. The mesh shown in (B) is created by using honeycomb 2 algorithm from the mesh shown in (A). The subdivided version of the rind shape in (C) is used to create image in Figure 2A.

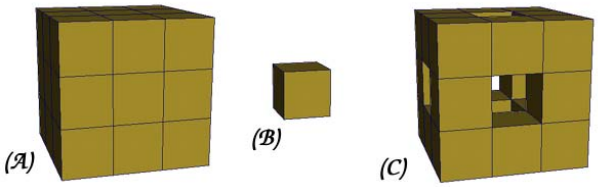


Figure 21. An example of limitations of semi-automatic approach. The offset mesh shown in (B) cannot be created without changing mesh structure of initial mesh shown (A). Users can easily create rind shapes in such cases as shown in (C).

[7] E. Akleman, J. Chen and V. Srinivasan, A Prototype System for Robust, Interactive and User-Friendly Modeling of Orientable 2-Manifold Meshes, *Proceedings of International Conference on Shape Modeling and Applications 2002*, (SMI'02), (2002), pp. 43-50, Banff, Canada, May 2002

[8] E. Akleman, J. Chen and V. Srinivasan, A Minimal and Complete Set of Operators for the Development of Robust Manifold Mesh Modelers, (extended version of [7] selected and submitted to special issue of *Graphical Models Journal*).

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

[10] 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.

[11] E. Catmull and J. Clark, "Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes", *Computer Aided Design*, **10** (September 1978) 350-355.

[12] J. Chen and E. Akleman, Topologically robust mesh modeling: concepts, structures and operations, Submitted to a journal. Technical report is available at www-viz.tamu.edu/faculty/ergun/topology.

[13] J. Chen, "Algorithmic Graph Embeddings", *Theoretical Computer Science*, **181** (1997) 247-266.

[14] D. Doo and M. Sabin, "Behavior of Recursive Subdivision Surfaces Near Extraordinary Points", *Computer Aided Design*, **10** (September 1978) 356-360.

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

[16] M. C. Escher, *The graphic works: introduced and explained by the artist*,, (image plate 41 and 43, Barnes and Nobles Books, New York, 1994.

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

[18] The Fast Light Toolkit Home Page: http://www.fltk.org/

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

[20] M. Forsyth, “Shelling and Offsetting Bodies”, *Proceedings of Solid Modeling’95*, pp. 373-381, Salt Lake City, Utah, May 1995.

[21] J. R. Rossignac and A. A. Requicha “Offset Operation in Solid Modeling”, *Computer Aided Geometric Design*, No. 3, pp. 15-43, 1985.

[22] 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.

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

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

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

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

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

[28] 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 .

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

[30] V. Srinivasan, E. Akleman and J. Chen, Interactive and user friendly construction of multi-segment curved handles, *Proceedings of Pacific Graphics 2003*, October 2003.

[31] W. Welch and A. Witkin, “Free-Form Shape Design Using Triangulated Surfaces”, *Computer Graphics*, **28** (August 1994) 247-256.

[32] D. Zorin and P. Schröder, co-editors, *Subdivision for Modeling and Animation*, ACM SIGGRAPH’2000 Course Notes no. 23, July, 2000.