Converting a Möbius-Torus Into a Walkable Sculpture

Brian Aronowitz
Carlo H. Séquin

Converting a Möbius-Torus Into a Walkable Sculpture

Brian Aronowitz and Carlo H. Séquin
CS Division, University of California, Berkeley
E-mail: sequin@cs.berkeley.edu

Abstract

This project started as a collaboration with Wanqin Nong, a New York City based artist. Inspired by Séquin's video *Topology of a Twisted Torus*, the artist created the concept of a room-sized walk-in *Möbius Torus* that could be explored on foot. Nong also specified that the torus should display an elegant elongated, egg-shaped geometry, with some important proportions expressing the golden ratio. In this report, we describe the techniques we used to construct such a geometrical model, which is suitable for both architectural visualizations and 3D-printing.

1. Introduction

This project started as a collaboration with Wanqin Nong, a New York City based artist [4]. Inspired by Séquin's video *Topology of a Twisted Torus* [6], the artist introduced the concept of a room-sized walk-in *Möbius Torus* that could be explored on foot. The geometry of this structure is essentially a solid torus minus a thick Möbius ribbon twisting through 180 degrees as it makes a full loop inside the torus. Visitors to this exhibit should be able to walk on all surfaces that do not deviate too much from horizontal – in the internal Möbius space, as well as on top of the toroidal structure. Nong also specified that the torus should display an elegant elongated, egg-shaped geometry, with some important proportions expressing the golden ratio. In this report, we describe the techniques we used to construct such a geometrical model, which is suitable for both architectural visualizations and for 3D-printing.

2. The *Möbius Torus*

To create such a *Möbius Torus*, two crescent-shaped profile curves (Fig.1a), which define the cross section of the solid volume of the *Möbius Torus*, are swept around a circle and twisted 180 degrees as they complete that sweep (Fig.1b). Creating mesh segments in between subsequent instantiations of this cross section results in a clean 2-manifold mesh (Fig.1c). This geometry was generated with Berkeley SLIDE [5] and visualized in Blender3D [1] (Fig.1d).

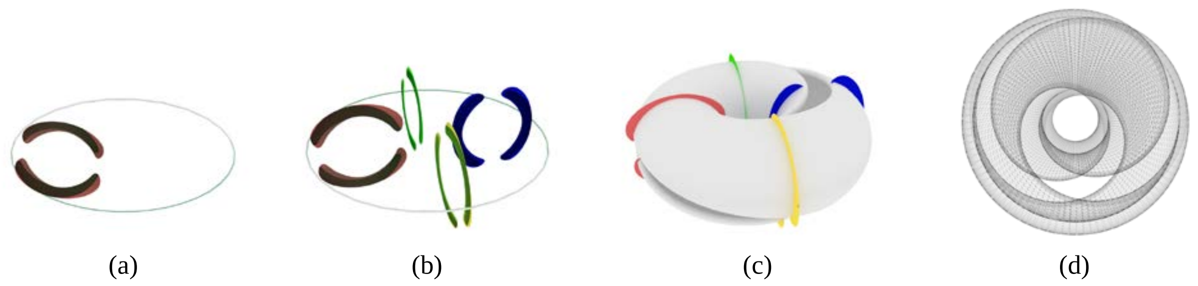
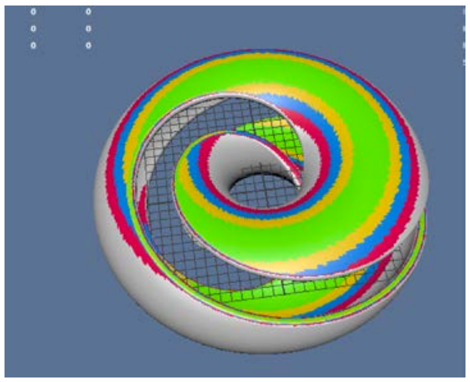


Figure 1: *(a) Single cross-section; (b) four cross-sections; (c) Möbius Torus plus four select cross sections (enlarged); (d) wireframe rendering of a basic Möbius Torus.*

3. Walkable Surfaces

With a model of the basic *Möbius Torus* in hand, the next task was to identify all the surfaces that could be made accessible to walk-in visitors. We decided to convert all surfaces, where the surface normals deviate less than 45 degrees from vertical, into staired regions. Figure 2 shows the *Möbius Torus* surface colored by zones of different slopes with respect to horizontal. Green indicates regions that deviate less than 15° from horizontal. Yellow, blue, and red, correspond to zones with slopes of {15° to 25°}, {25° to 35°}, and {35° to 45°}, respectively. The non-walkable surfaces, shown in grey, will be maintained as a smooth mesh.


(a)

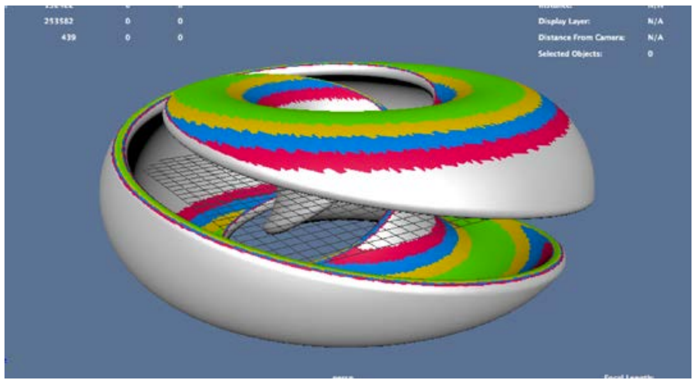

(b)

Figure 2: *Surface regions of different slopes on the Möbius Torus: (a) top view, (b) side view. Green: (0° to 15°), yellow: (15° to 25°), blue: (25° to 35°), red: (35° to 45°).*

4. Stair Generation Algorithm

The next task was to develop an algorithm to insert well-formed stairs into the model. The main challenge here was to blend the edges of all the stairs nicely into the smooth mesh surrounding them, so that overall, we still maintain a water-tight boundary representation.

Edge Insertion

The process begins by slicing the *Möbius Torus* into 50 horizontal slices, all 7 inches tall, which corresponds to a comfortable stair step height (Fig.3). In the stair zones, the intersection edges created in this slicing process will then become the edges of individual stair steps. To create these slices, we used the *Bisect_Plane* function built into Blender3D [2]. We labeled these edge location "Z1" to "Z50," since in the context of this work, the vertical direction is represented by the Z-axis, while the XY-Plane corresponds to the horizontal (equatorial) plane.

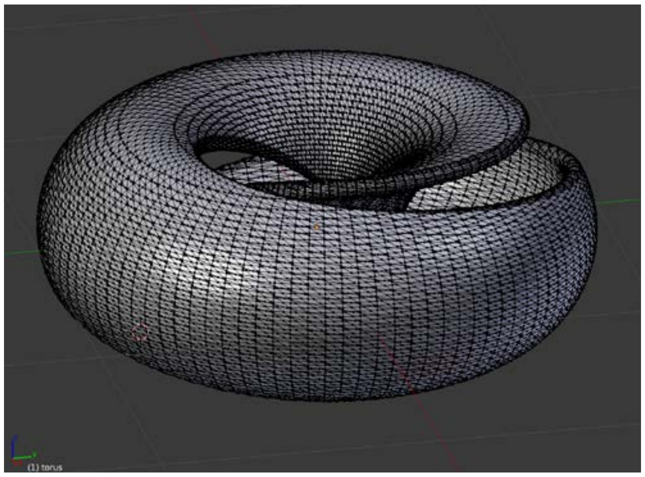
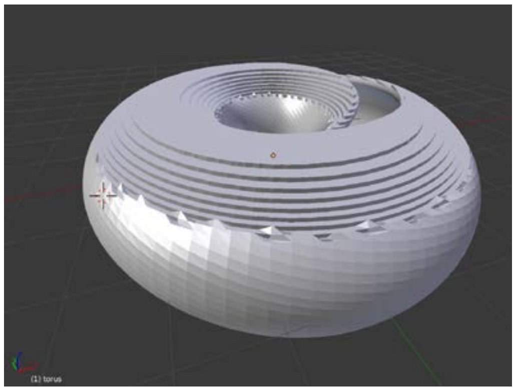


Figure 3: *Möbius Torus sliced into 50 horizontal slices.*

Stair Projection: The Upper and Lower Surface Areas

Initially, our approach to stair generation was simply to "stair-project" downwards all facets with face normals whose Z-angles were less than 45°. In this case, "stair projection" means setting the Z-coordinates of relevant vertices of mesh facets in the stairs areas downwards onto the stair-Z location directly below them. However, since the original toroidal mesh facets were not naturally aligned with the edges produced in the slicing process, facets that partly overlap the stairs areas, as well as the remaining smooth mesh surface areas need some special treatment. Otherwise, such facets will lead to erratic geometry in the border regions (Fig.4a).


(a)

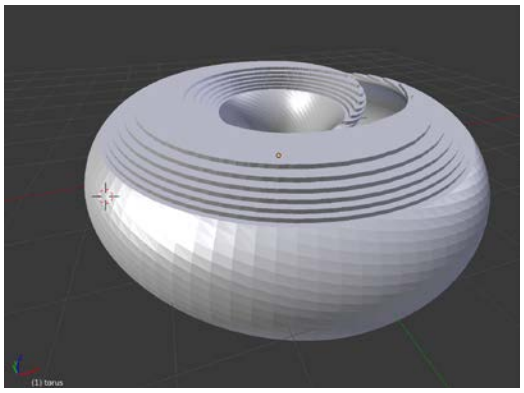

(b)

Figure 4: *Problems at stair boundaries: (a) simple approach, (b) using consensus algorithm.*

To fix this, we implemented a *consensus algorithm* relying on "bins" in the Z-range of possible values between stair edges to determine whether all the vertices within such a bin should be projected downwards. The *consensus algorithm* then iterates through all facets in the mesh and counts their vote towards their bin. The face bin is determined by the Z-coordinate of the centroid of its component vertices: and it belongs to the bin that it falls in, lower-bound inclusive. As the algorithm counts, every facet with a slope less than or equal to 45° would contribute a 'yes' to the projection vote; steeper facets would contribute a 'no'. After the votes are tallied for each bin, if over 50% of the facets have voted for projection, the entire bin is turned into a stair step. This approach creates cohesive stair steps with smooth boundaries, and it is aesthetically satisfying (Fig.4b).

The smoothness of the consensus algorithm did come with a cost: it prevents any stairs from being created on the narrow ascending and descending ridges of the torus. Therefore, we added a second phase to the stair algorithm. At the end of the *consensus algorithm*, we denote the boundaries where it has stopped producing stairs as E1, for the upper edge, and E2 for the lower. These are used in the second phase of the algorithm.

Stair Generation: Defining the Stair Areas on the Ridges

On the highly curved ridges ascending from the lower walkable surfaces on the inside of the torus, to the upper platform on the outer surface of the torus, we needed a more sophisticated metric than just the angles of the normal vectors to decide which facets to flatten. To define the regions on the curved ridges that will be converted into walkable stair, we relied on user-specified boundary curves, which then defined stair areas with smooth boundaries.

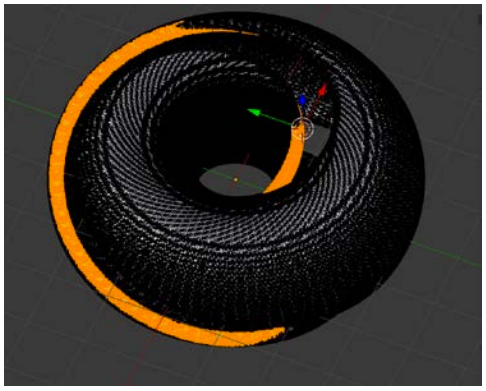


Figure 5: *Visualization of potential smooth stair region.*

To delineate the ridge stair areas, we started with two user-selected edges on the ascending ridge, as shown in Figure 6a. For each initial edge, the algorithm iteratively selects the connected edges that have a maximal dot product with the previous edge. By iteratively running up and down the edges, our maximum dot product heuristic selects the edges that capture the smooth shapes of the ridge curves. The algorithm iterates until it hits boundaries E1 and E2. At these top and bottom bounds, the algorithm then selects the edges that have the lowest dot product with the ascending curve: (finding the most orthogonal edge), and it then uses, once again, the maximal dot product metric, but with the additional constraint that the Z-coordinates of the selected edges must fall on either boundary E1 or E2. This heuristic forces the algorithm to select edges that follow the smooth curvature of the mesh, which creates a smooth boundary curve for us to project with. Running this algorithm, starting from four initial boundary selections, builds a smooth, fully closed boundary loop (Fig.6b). The Blender3DAPI function *Loop_To_Region* [3] then creates a mesh that fills in this boundary loop (Fig.6c).

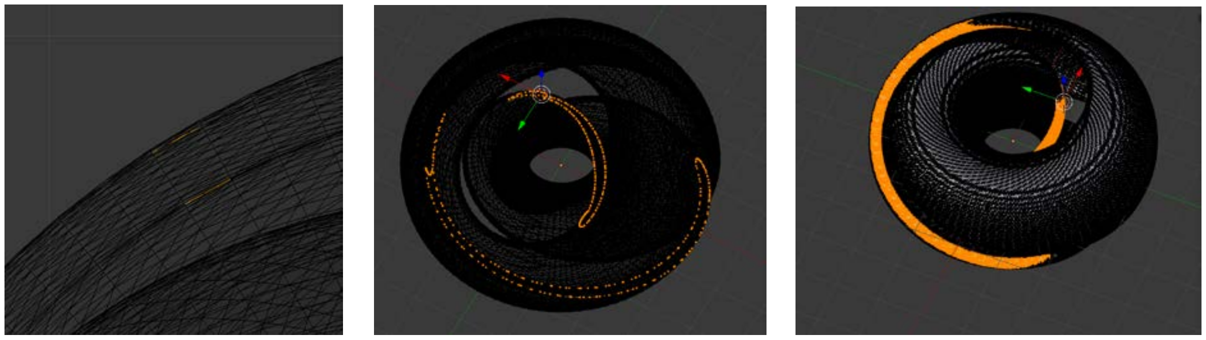

(a) (b) (c)

Figure 6: *(a) Initial selection; (b) generated smooth boundaries; (c) filled-in boundary loop.*

Stair Projection in the Ridge Areas

Once the stair areas on the ridges had been defined as described above, we then followed the approach used in the larger walkable surface areas, projecting all vertices in these areas down to the stair level below them. Unfortunately, this naive projection left bad artifacts where stair faces intersected with facets in the smooth meshes nearby (Fig. 7a).

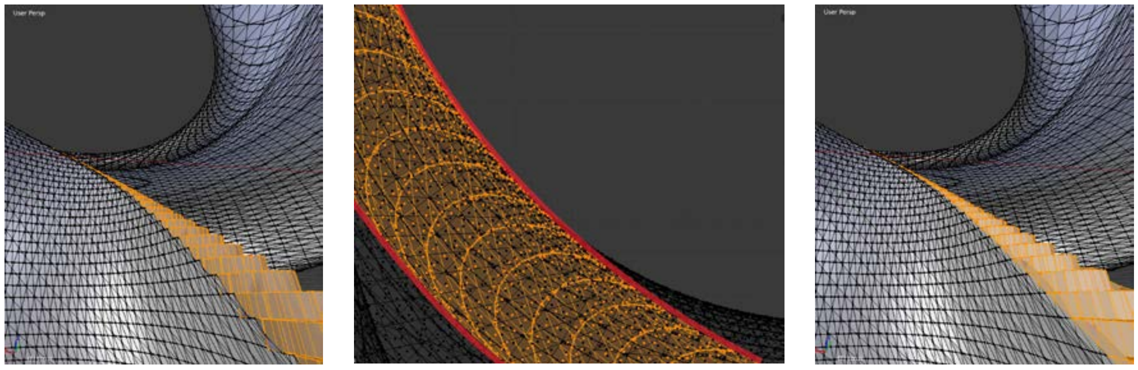

(a) (b) (c)

Figure 7: *(a) Stair-projection problem; (b) boundary visualization; (c) fixed intersection artifacts.*

These intersections resulted from vertices near the boundary curves that were projected downwards through overhangs beneath them (Fig. 7a). These overhangs originate when non-projected vertices in a stair bin have an XY plane position that lies within the bounds of the boundary curves. When some of these vertices are projected downwards, they intersect with the faces in these overhangs. To prevent these bad projections, we introduced a constraint for our original algorithm that uses the boundary edges to constrain our stair projections (Fig. 6b, Fig.7b). Before projecting a vertex, the algorithm checks whether the XY position of the vertex lies within a threshold distance of its nearest inner and outer boundary vertices. A vertex is only projected down to the lower stair level if the following constraint is met:

Constraint: (c.xy - v.xy) > b1(c.xy - b_inner.xy) and (c.xy - v.xy) < b2*(c.xy - b_outer.xy)*

Where .xy are the x,y coordinates of the vertex, ***c*** *is the centroid of the mesh;* ***v*** *is the vertex being projected;* ***b_inner*** *is the closest inner boundary point;* ***b_outer*** *is the closest outer boundary point;* ***b1*** *= 1.1 and* ***b2*** *= 0.9.*

This constraint effectively prevents self-intersections by bounding the projections to a smaller window; it confines the projected vertices to areas that do not sit above overhangs. (Results shown in Fig. 7c).

5. Final Shape Adjustment

After creating a watertight boundary description with stairs that transitioned nicely into the smooth mesh, we dealt with the requirement of producing an egg-shaped torus where the ratio of the half-axes in the X- and Y-directions corresponded to the golden ratio φ=1.618.... After experimenting with several linear and non-linear functions, we found that a simple linear stretch on half the vertices could match the artists drawings exactly. Thus, the half-circle with positive Y-coordinates is stretched into an elongated half-ellipse.

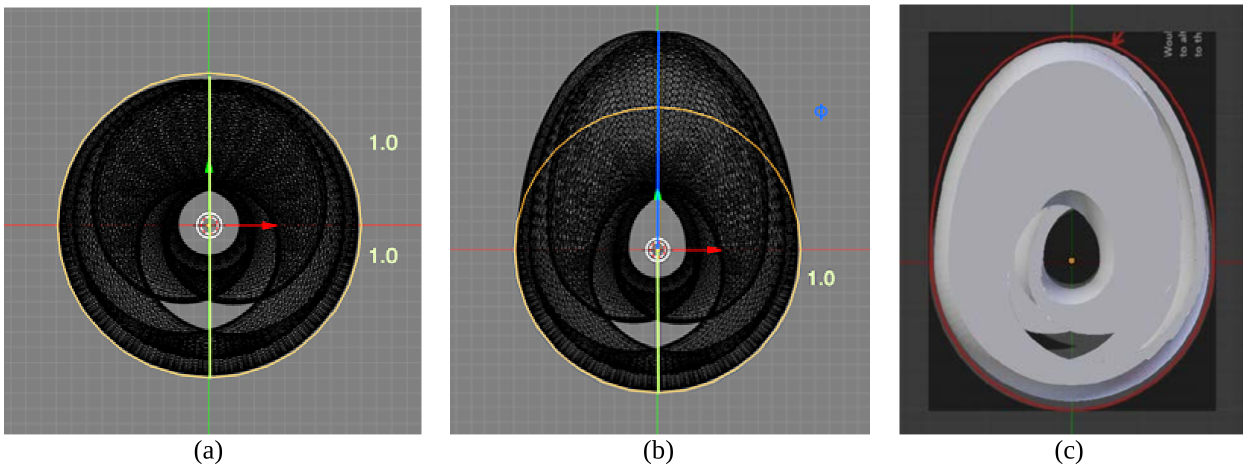


Figure 8: *Stretch function: Y' = Y * φ, for Y > 0, with φ = 1.6180339*

(a) Original torus shape; (b) upper half of torus stretched by a factor of φ; (c) elongated model.

6. Final Visualizations

Figure 9 shows the final re-shaped *Möbius Torus*. It shows the smooth inner surfaces in red; the smooth outer surfaces in yellow, the stair areas in blue, and in green the boundary zones between the two domains, where special processing was necessary. This was rendered using the *Cycles Pathtracer* built into Blender3D [2].

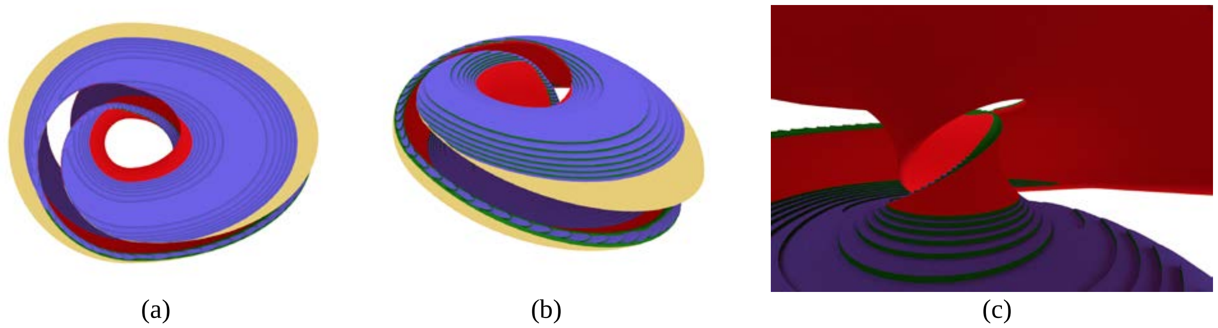


Figure 9: *Final scaled Möbius Torus model: (a) top view, (b) side view, (c) an inside view.*

Red: smooth inner surfaces; yellow: smooth outer surfaces; blue: stair areas; green: boundary zones.

Final 3D prints:

To get a better sense of what a finished architectural model would look like, and to test the consistency and water-tightness of the generated boundary representation, we ran our model through Cura Type A software and 3D printed it in PLA plastic (Fig.10).

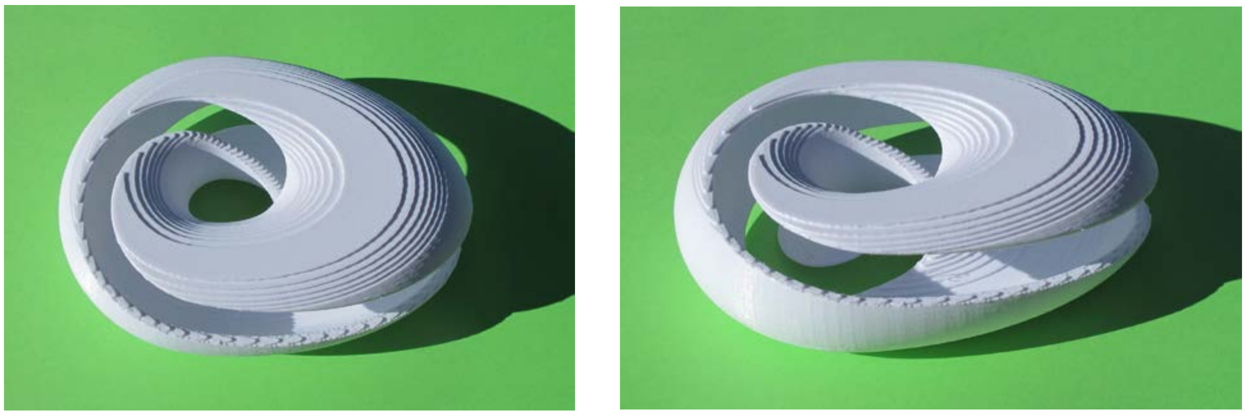

(a) (b)

Figure 10: *A detailed 3D print: (a) view close to top, (b) view more from the side.*

7. Summary and Conclusions

Our goal was to make a geometrical demonstration model for artist Wanqin Nong, based on which she could develop a large-scale structure with walkable stair-shaped surfaces. The most challenging part in this task was to turn the surface areas that were only moderately sloped into stepped regions with smooth edges and with pleasing transitions from the stepped areas into the steeper, smooth surface regions. The final 3D model proves that our approach was properly meeting this challenge.

More challenges remain to make this into a large-scale, walkable structure. The stairs on the ridges may be too narrow even for single-file foot traffic, and certainly for two pedestrians passing by one another in those areas. Thus, an additional transformation may have to be introduced to widen these stairs on the curved ridges of the toroidal structure. Moreover, practically everywhere, where stepped regions transition into the steeper smooth surface areas, safety railings will have to be introduced.

Moreover, there are artistic and architectural challenges in the selection of the best textures and colors for the steps as well as for the walls. The goal is to make this structure look like an attractive monument from some distance away, and also like a warm and welcoming environment once a visitor steps inside.

References

[1] Community, B.O., 2018. *Blender - a 3D modelling and rendering package,* Stichting Blender Foundation, Amsterdam. – http://www.blender.org.

[2] Community, B.O., 2018. *Blender - a 3D modelling and rendering package,* Stichting Blender Foundation. – https://docs.blender.org/api/blender_python_api_2_77_3/bmesh.ops.html?highlight=bisect_plane#bmesh.ops.bisect_plane

[3] Community, B.O., 2018. *Blender - a 3D modelling and rendering package*, Stichting Blender Foundation. – https://docs.blender.org/api/2.79/bpy.ops.mesh.html?highlight=loop_to_region#bpy.ops.mesh.loop_to_region

[4] W. Nong, Art Director and Visual Problem Solver: home page; – https://www.nongwanqin.com/bio-1

[5] J. Smith: “SLIDE design environment.” (2003). – http://www.cs.berkeley.edu/~ug/slide/

[6] C. H. Séquin: “Topology of a Twisted Torus”. – https://www.youtube.com/watch?v=3_VydFQmtZ8&feature=youtu.be