Combining Procedural Modeling and Interactive Graphical Editing for the Design of Abstract Geometrical Sculptures

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

Abstract

We describe an experimental CAD tool aimed at creating intricate 2-manifold surfaces suspended by complex tangles of border curves. This exploratory system aims at combining the best of two worlds: high-level procedural modeling for capturing various symmetries and replications of geometrical entities, and interactive graphics editing to define the surface elements that would be difficult to specify procedurally in an initial design file. The key issue under investigation is how to best capture the geometrical changes made in the graphics domain and reintroduce them into the procedural design file, which, in the end, is supposed to define the whole sculpture, while maintaining the flexible parameterization contained in the original specification.

1. Introduction

Many of the abstract geometrical sculptures by artists such as Naum Gabo [7], Eva Hild [9], or Charles Perry [13] define intricate 2-manifold surfaces suspended by complex tangles of border curves (Fig.1). We found it very difficult to model such geometries with commercial or public-domain CAD tools such as 3ds Max [1], Maya [11] or Blender [2].

(a) Soap film (b) Naum Gabo [7] (c) Eva Hild [9] (d) Charles Perry [13].

Figure 1: *Soapfilm-like surfaces suspended on smooth border curves.*

Creating the many interlinked border curves, while maintaining perfect symmetry, seems almost impossible with a typical on-screen, interactive, graphical modeling tool. It is much more convenient to define these curves in a procedural manner, for instance with a set of properly parameterized B-spline curves. These parameters can then be used to fine-tune the border curves as well as the suspended surface(s), while inspecting the complete sculpture in its (almost) final form.

On the other hand, it is even more difficult to define all the surface elements in their final form procedurally, that is, individually specifying the set of face vertices needed for all surface facets in code or some other user interface. For this part of the design task, an interactive graphics interface is much more appropriate for selecting suitable segments on some of the border curves and defining some surface patches between them. Most of the above mentioned, commercial CAD tools offer some ways of connecting curve segments with "bridges" or by extrusions. But, typically, these surface elements are captured in some low-level, non-hierarchical form and cannot be integrated into the original hierarchical, procedural description of the border curves. Some of the tools even force the user to convert the parametric description into a flat mesh before any mesh-level edits can be made. This is a particularly severe shortcoming, when the sculpture

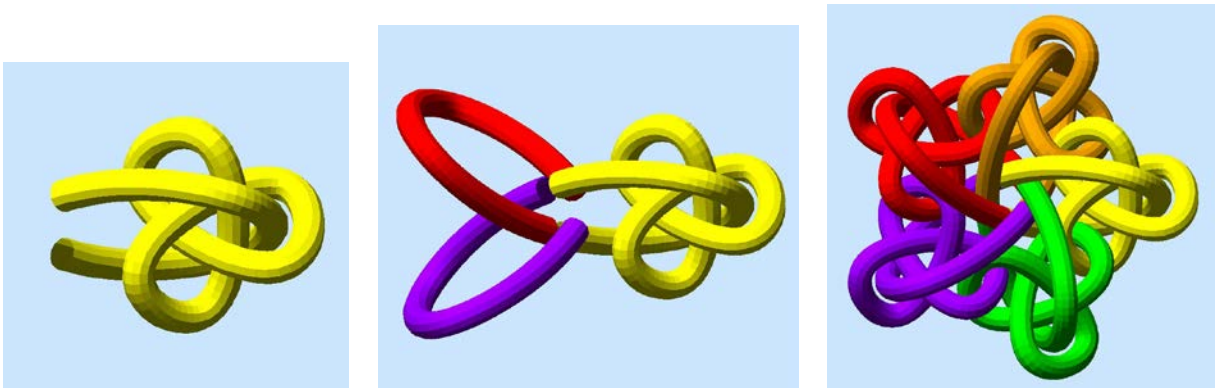
under construction exhibits a lot of symmetry, such a Perry's *Star Cinder* [14] (Fig.6a). In a sculpture like this, with 60-fold symmetry, it should be sufficient to define only $1/60^{th}$ or $1/30^{th}$ of the overall surface and then let the procedurally specified symmetries construct the remainder of the surface. For this purpose, it is desirable to have the interactively added parts integrated into the original high-level description. If the needed prototypical surface elements are placed into the proper location in the procedural description, all necessary replications of these elements will happen automatically.

With the specific goal of reconstructing sculptural surfaces like the ones created by Eva Hild (Fig.1c) [9] or by Charles O. Perry (Fig.1c) [13] or (Fig.6a) [14], we have started the development of a CAD tool, called NOME (Non-Orientable Manifold Editor), that helps us to create and optimize such designs. Our goal is to combine the best features of high-level procedural modeling for capturing various symmetries and replications of geometrical entities, with direct interactive graphics editing for specifying needed surface elements. The key challenge is to let the graphical editing step not be the final design phase, but to re-integrate these edits into the initial procedural file so that it now specifies the sculpture in its current form. The designer should be able to pick up this file at a later time and continue the editing and optimization of the geometry, while still enjoying the full functionality of all the various parameterizations that were included in the original specification. Achieving this goal is surprisingly difficult, and it accounts for the fact that this development has stretched over the last three years [6][19]. While the evolving NOME tool has been good enough to design the intended complex geometrical 2-manifolds, as demonstrated by some examples later in the paper, the integration between procedural descriptions and graphical editing is far from being solved perfectly. This paper tries to bring some of the pending issues to the forefront and describe our current thinking about them.

2. NOME Files

NOME is a design-specific language, akin to that of a single-assignment programming language. It follows the normal paradigm of most hierarchical graphics scene descriptions. It defines *points* by their (*x*,*y*,*z*) coordinates, and then uses such points to define *edges*, *polylines*, *faces*, and *meshes*. *Instances* of several such elements can be combined in a *group*. Instances of such groups can then be placed in multiple locations in the scene, after suitable optional transformations, including *non-uniform scaling*, *rotation around an arbitrary axis*, and *translation*. For instance the 5-fold rotational (C_5) symmetry around the chosen (0 0 1) *z*-axis, exhibited by the model shown in Figure 2c, is expressed with the following statements:

```
instance   ID_0   MODULE  yellow      end_instance
instance   ID_1   MODULE  orange   rotate(0 0 1)(72)   end_instance
instance   ID_2   MODULE  red      rotate(0 0 1)(144) end_instance
instance   ID_3   MODULE  purple   rotate(0 0 1)(216) end_instance
instance   ID_4   MODULE  green    rotate(0 0 1)(288) end_instance
```



(a) One MODULE *(b) connections between neighbors* *(c) five MODULEs joined*

Figure 2: *Cinquefoil with five interlinked trefoils.*

Of course, "MODULE" (Fig. 2a) may itself be a hierarchical construct, consisting of instances of other, smaller groups; for instance, MODULE could contain one complete trefoil and 1/5th of the cinquefoil:

```
group MODULE
  instance  ID_5  One-fifth-of_cinquefoil  end_instance
  instance  ID_6  Complete_trefoil-knot   end_instance
end_group
```

It may be difficult to define the exact geometry of MODULE (Fig.2a) up front in a procedural manner. The cinquefoil cannot be completed (Fig.2b), nor the trefoil properly dimensioned and positioned, until they can be judged in the context of the complete 5-fold assembly (Fig.2c). Thus, it is desirable to specify the MODULE geometry with some built-in parameters, which may concern non-uniform scaling, its placement in 3D space, or the details by which the cinquefoil loop in MODULE is connecting with its two neighbors (Fig.2b) to form, in the end, the complete cinquefoil.

The key mechanism for parameterization is through alteration of any numerical value in the NOME specification. Wherever a numerical parameter is needed for a NOME statement, e.g. to specify the *x*-, *y*-, or *z*- coordinates of a vertex, or the radius, *r*, of some circle, an expression must be supplied. This can simply be a number, or it may entail operations involving complex algebraic expressions and several numerical variables that can be adjusted interactively with corresponding *sliders* [6]. Whenever the current state of design is saved, the latest values of these sliders are also saved, and the necessary updates are made in the initial procedural description.

If MODULE, and all other geometry, are defined procedurally, and thus the whole design emerges from a procedural text file, this approach presents no real problems, and can be realized within many CAD environments, ranging from Rhino3D [15] to Berkeley SLIDE [16]. Difficulties arise, when a designer wants to introduce new elements into the design that are difficult to specify in the initial procedural description, such as some surface elements between arbitrary segments of different border curves. Such facets are more naturally introduced by clicking on sample vertices along some of the border curves. The challenge is to find a way to integrate these interactively defined design extensions into the original program that was used as the start of the design.

Many computer-aided design or modeling tools combine some procedural input with subsequent direct manipulation in a graphics display. Only a few tools try to integrate the changes made in the interactive graphics environment into the original procedural description. Most advanced in this respect is the *Sketch-n-Sketch* system [5][8], where a generator program is constructed based on the user's graphical drawing and editing operations. For our current application, where we want to construct complex 3D object with high symmetry, rather than 2D vector graphics designs, we always start with a procedural description that captures the basic structure and all its symmetries; normally this framework is not changed interactively.

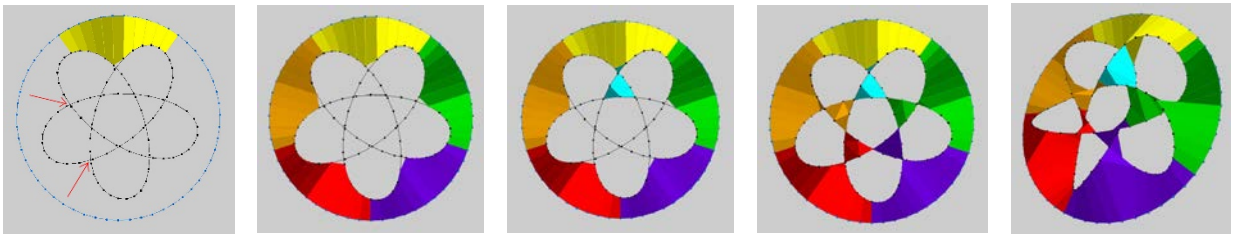
The designer typically starts by specifying a symmetrical tangle of border curves or a set of strategically place anchor faces that mark the main symmetry axes. Then, additional faces are added to define the topology of the final surface. NOME is crafted specifically to support the capability of constructing stretchable surfaces between procedurally defined geometrical features. This capability of linking up an arbitrary collection of vertices to form a stretchable face is made possible by assigning every geometrical entity, down to the vertices at the leaves of the scene tree, a unique hierarchical identifier. Every *vertex*, *object*, or *instance-call* is given an *ID*, i.e., a unique identifying label. The name for an arbitrary vertex then is constructed by concatenating all these labels along the path from the root of the scene tree to its final leaf instance. For vertices created by a built-in generator, such as the *torus* generator, or the *B-spline* generator used to define border curves, the last part of the vertex name is produced inside the corresponding generator. New facets, generated in an interactive, graphical editing session, are then described by reference to these composite textual labels, and can thus be added and referenced unambiguously in the original NOME design file.

A few higher-level operations make it easier to connect larger segments of border curves. Two multi-segment regions of border curves can be selected, and a zippering operation can then be invoked that forms a bridging ribbon between these two borders, composed of quadrilaterals and possibly some extra triangles, that prevent the connecting quadrilaterals from becoming too skewed [6].

In general, we want to save all changes that a designer makes in the interactive graphics environment in the initial procedural specification, while changing this file as little as possible. When a designer reopens this file at a later time to make further design enhancements, the file should look as familiar as possible. At intervals convenient to the designers, we let them save the state of a design. At those times, the most recent changes made are saved as an appendix to the original NOME file. Typically, the designer then reopens this file and moves some of the newly created facets to an appropriate place in the hierarchical description, so that these facets are now replicated with the symmetries specified in the original design; the displayed surface will then reappear more fully instantiated. This switching back and forth between two quite different views of the emerging design has proven quite powerful and convenient for creating the type of intricate 3D models presented below.

3. NOME Workflow Example

This section presents a simple tutorial example to describe a typical workflow in NOME for the creation of a soap-film-like surface spanning a cinquefoil surrounded by a circular ring (Fig.3). The border curve of the cinquefoil is specified as a cubic B-spline with three geometry parameters that makes it easy to adjust the extent and the width of the lobes, as well as the closest distance between the knot curve and its 5-fold symmetry axis. There is also a parameter that describes the amplitude of the undulation in the *z*-direction and thus controls the separation of the knot curve branches at the locations of the projected knot crossings (red arrows in Fig.3a). In the shown example, the B-spline is sampled at 90 uniformly spaced parameter values, and the circle is represented as a 55-sided polygon; both of these values are tied to sliders and can be changed interactively. These particular values are higher than they need be to define the final 2-manifold well enough so that it looks good after a few levels of refinement by Catmull-Clark subdivision [4]; but they provide enough clickable locations so that the designer can readily insert facets with a variety of shapes and aspect ratios. The designer need not make use of all the available sample points on the border curves.



(a) *(b)* *(c)* *(d)* *(e)*

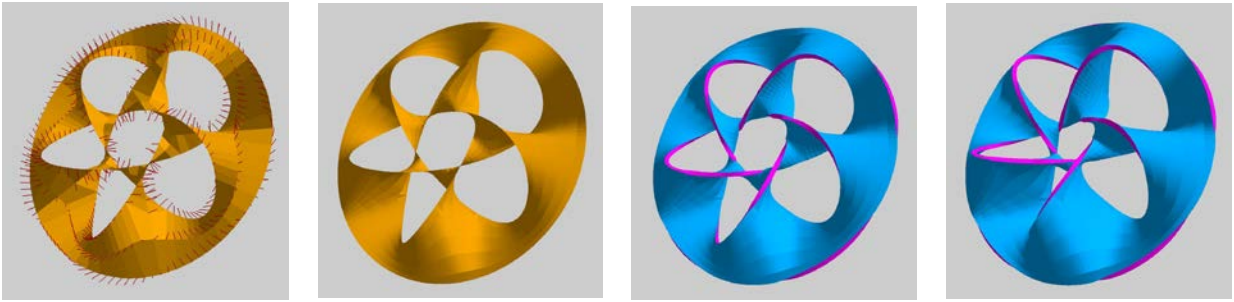
Figure 3: *Incremental construction of a 2-manifold suspended by the given two border curves.*

Another consideration in choosing the sampling density on each curve concerns the formation of nicely twisted saddles at all points where there are projected knot crossings. Ideally, one would like to have a pair of sample points as close as possible to the point of minimal distance in the skewed crossing of the two curves (red arrows in Fig.3a). Finally, the number of sampling point should respect the symmetry of the emerging design. The sample vertices must lie in the same respective locations on all the corresponding lobes of a given knot, so that any partial geometry that is subjected to the specified symmetry operations will have its vertices land on corresponding vertices on the target lobes. This is important in order to obtain a clean merger between the various partial geometry pieces, entered manually or generated automatically, into a single cohesive 2-manifold.

With a good choice of sampling points, the designer can now click on sets of (typically four) vertices and define them as facets in the forthcoming surface (Fig.3a). Quadrilaterals are the preferred facet shapes,

when the surface is later refined with Catmull-Clark subdivision [4]. A good start is to place the most clearly needed facets on the inside of the outermost rim, connecting them to the dominant lobes of the cinquefoil knot (Fig.3a); these facets then form parts of a continuous outer ribbon that will eventually travel around the whole sculpture. When roughly one fifth of the circumference has been filled in, it is time to save the current state of design. In the saved NOME file, a list of the newly added facets will appear as an appended small new mesh plus an *instance* call to that mesh. In this text file, the designer can now readily add four more instance calls with different rotation angles around the *z*-axis, in increments of 72°, as in the example in Section 1. Giving the five instances different colors, makes it easy to see, whether the added facets properly compose the whole circumference ribbon (Fig.3b), whether there is still a gap to be filled, or whether some of the facets in the different instances already overlap. Those flaws are then easily corrected, and the designer can resume an interactive editing session based on the modified NOME file.

The next step is to fill in the inner ring of five "triangular" regions (Fig.3c). Newly added facets can now either be included into the ribbon mesh created in the first editing session, or they can form a separate mesh that is replicated five times on its own, and which may be colored differently (e.g., cyan). The latter approach allows one to temporarily turn off some or all of the instances of the first mesh, to enhance the visibility into the interior of an emerging sculpture. This is not much of an issue in this 2.5D example, but becomes important in the complex, nested, 3D geometries discussed in Section 4. Eventually, all the facets have been placed, and a complete hierarchical specification of the desired 2-manifold has been generated (Fig.3d). Figure 3e provides an oblique view of this same assembly to yield a better understanding of the 3D nature of the twisted saddle geometries emerging at the border-curve cross-over points. The NOME-code describing the state of design shown in Figure 3d can be found on-line [17].



(a) (b) (c) (d)

Figure 4: *Smoothing and offsetting. (a) subdivision=2 with normal vectors; (b) subdivision=3; (c) offset surface generation; (d) adjusting some geometry parameters.*

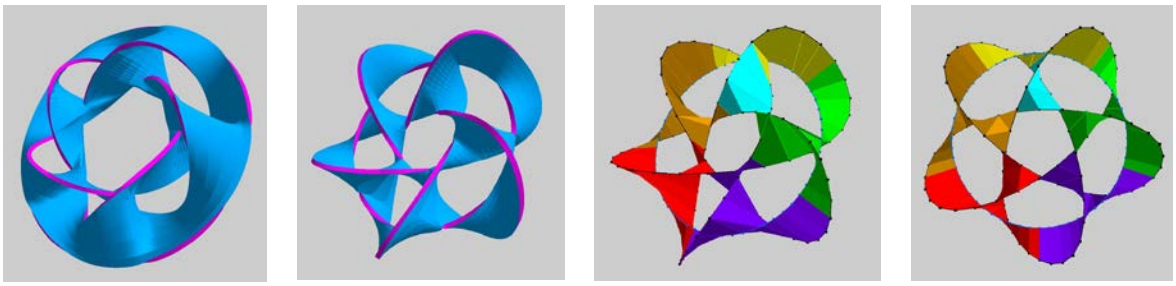
Now it is time to do a *merge* operation. In this process, in a bottom-up, recursive manner, all singly-used *border edges* of local pieces of the 2-manifold are tested for coincidence. If both pairs of end-vertices of a pair of border edges are within some small *epsilon*-distance of one another, these vertices and the edge in between will be joined. Here the initial, precisely defined, 5-fold symmetrical, procedural specification contained in the original NOME file plays an important role. It guarantees, that the vertices of the few prototype faces added graphically in one fifth of the sculpture, after replication and transformation, will align properly with the border-curve vertices in the other four sectors of the sculpture, so that the merger can be carried out without any problems.

An efficient, hierarchically flat, winged-edge data structure is now constructed to represent the emerging 2-manifold (Fig.4a,b). This data structure is used subsequently to perform mesh-refinement through two to four steps of Catmull-Clark subdivision [4]. This finely tessellated surface can be shown with or without surface normal vectors displayed at all vertices. This is the moment to inspect visually the generated surface for proper connectivity and for any undesirable spots with unacceptably high curvature. Furthermore, as long as we rely simply on Catmull-Clark subdivision for surface refinement and smoothing, the way the designer selects quadrilaterals affects the final geometry of the surface. Constructing long and narrow quads with an aspect ratio of 3 or more, will typically hamper the formation of a nicely balanced

saddle as one would find in a perfect minimal surface. Such flaws will show themselves at this stage and can be ameliorated by selecting facets with lower aspect ratios and by slightly changing some of the geometrical parameter values.

If the shape of the surface is satisfactory, the designer can now further smooth the surface with additional steps of subdivision and subsequently give it some "physicality" by constructing (blue) offset surfaces on both sides of the smoothed 2-manifold (Fig.4c). Along the border curves, the two surfaces are connected to one another with strips of small (red) quadrilateral facets.

Now there is a final opportunity to optimize the aesthetic appeal of the sculpture in its completed form. Perhaps the *z*-undulation amplitude of the knot may be enhanced to give the whole design more of a 3D character than was present in the earlier 2.5D configuration, which was more convenient for initial surface construction. A larger undulation (Fig.4d) will in general increase the distances between crossing border curve segments and soften the saddle geometries in the regions of the projected knot crossings. Being able to still change the geometric parameters of the knot curves, permits the designer to resolve some specific local trouble spots, as may be caused by two thickened surface elements getting too close to one another, or some saddles becoming too twisted. It is the responsibility of the designer adjusting the various sliders to watch out for unintended, bad side-effects, where solving one local problem will cause another one somewhere else. In Figure 5a the inner hole of the knot has been enlarged dramatically; but this now causes the inner five branches of the border curve to pass through the surface itself. This is not acceptable for the current design where we want to obtain an intersection-free 2-manifold.



(a) (b) (c) (d)

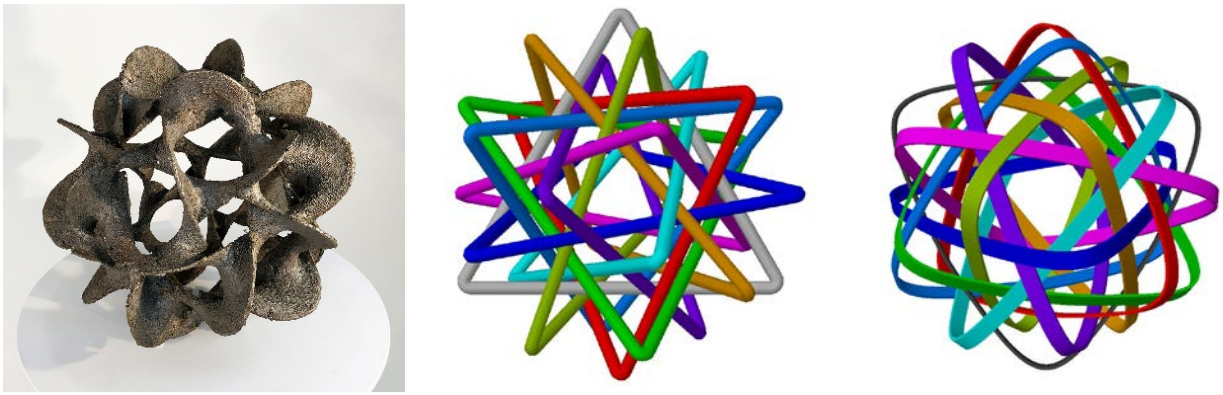
Figure 5: *Parameterized topological changes: (a) excessive enlargement of central hole; (b) topology change due to a reduced rim radius; (c) hierarchical view of new topology; (d) more geometrical changes: narrowing the cinquefoil lobes.*

In general, these adjustment operations are amazingly robust. In Figure 5b, the radius of the outer rim circle has been reduced so much that it now travels through the lobes of the cinquefoil knot. This forces this outer ribbon, which before was a gently undulating annulus, to transform into a twisted ribbon with five full twists. However, the result is still a proper 2-manifold without any self-intersections. At this point, the designer can readily switch back to a hierarchical view of the current design (Fig.5c,d) and possibly start another editing session to make further topological or geometrical changes.

4. A More Challenging Design

Here is a more complex 3D design that uses the power of NOME to the fullest. It was inspired by Charles O. Perry's *Star Cinder* [14] (Fig.6a). It is based on an *Orderly Tangle* [10] of 30 equilateral triangles with (oriented) icosahedral symmetry (Fig.6b). In Perry's sculpture, the pointy triangles have been softened into smoother triangular loops (Fig.6c), and a soap-film-like surface has been draped over the whole assembly of ten interlinked border curves. An inspection of this geometry shows that all the surface geometry is concentrated in a spherical crust in the outermost 30% of the circumsphere of this assembly. Our goal was to create a new generation of "Star Cinders" with a 2-manifold that will fill a larger fraction of the volume of the sphere, by forming several nested and tightly coupled spherical shells. This first requires more border

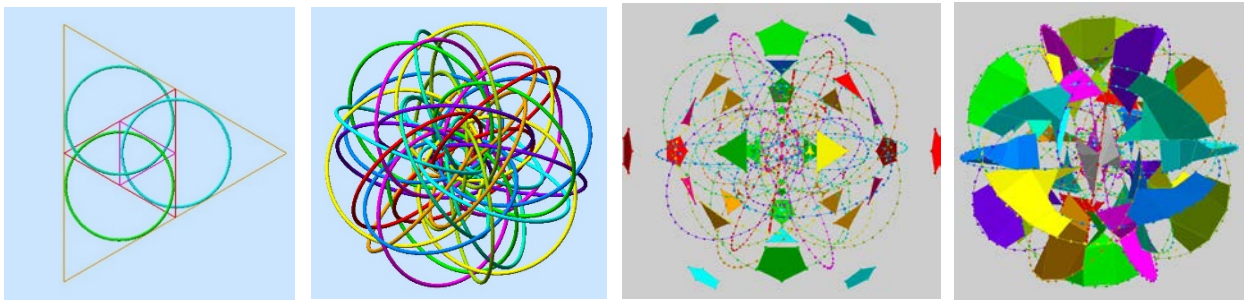
curve segments closer to the center of the sphere. This can be achieved by replacing all triangular loops with more complicated knots with 3-fold symmetry. A fist experiment was done with simple trefoil knots. An even denser tangle of border curves near the center emerges when the trefoil loops are replaced with (3,3)-torus knots, corresponding to sets of 3 symmetrically interlinked circles (Fig.7a).



(a) (b) (c)

Figure 6: *(a) Perry's "Star Cinder;" (b)"0rderly Tangle" of 10 triangles; (c) smooth triangular loops.*

These "*Triple-Loops*" are placed with the same icosahedral symmetry, and they now provide many border curve segments near the center of the sphere. Because of the high degree of symmetry exhibited by the 30 circles (Fig.7b), the pay-off from a hierarchical representation is correspondingly high. Each circle is associated with three geometric parameters (its *radius*, its *tilt-angle* with respect to the symmetry plane of the torus knot, and its *distance* from the center); they are adjusted to avoid any direct curve intersections and to maximize the separation of neighboring border curve segments. The challenge now is to create an intersection-free 2-manifold that uses all curve segments as border loops and maintains the overall icosahedral symmetry. To guide the designers in this task, triangular and pentagonal markers have been placed on the corresponding symmetry axis of the whole assembly (Fig.7c).



(a) (b) (c) (d)

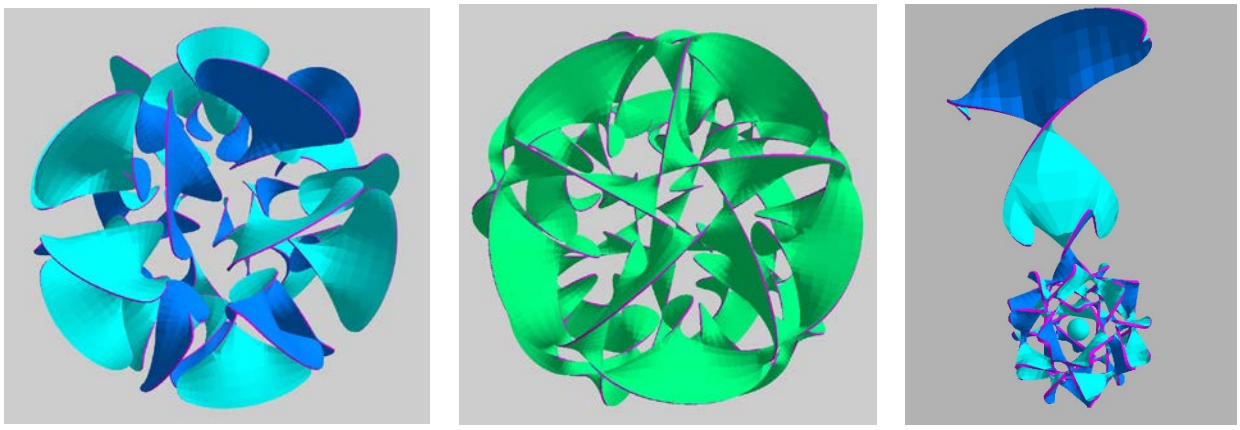
Figure 7: *(a) One Triple-Loop; (b) 10 Triple-Loops; (c) sampled NOME curves; (d) added facets.*

A good way to start manifold construction is by filling in the 30 clearly visible outer lobes by connecting the outermost curve segments in a mostly radial direction to the next inner curve segments encountered (Fig.7d and 8a). This explicit, manual construction needs to be done for only one single lobe, if the symmetry operations among the edges of an icosahedron have been captured properly in the transformations expressed in the 30 instance calls to the primary lobe. Pretty soon, a complete outer framework of twisted ribbons emerges (Fig.8b). On this framework, it is not too difficult to see where radial connections to the inner shells can be made (Fig.8c).

The inner portions are more difficult to connect properly; the tangle of border curves here is much denser; the various curve segments are always seen against much more cluttered surroundings; and the outer

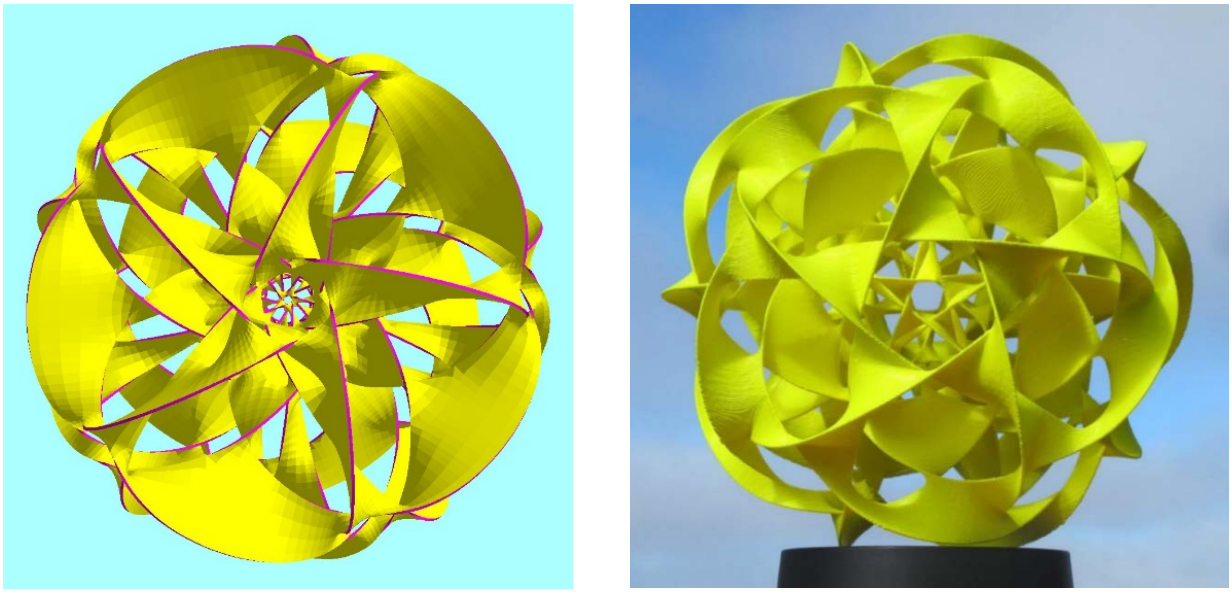
framework blocks visibility to the locations where new facets need to be added. Here the hierarchical procedural description provides some convenient help. Some of the listed instance calls that generate the 30-fold symmetry, or some of the facets in the basic mesh segment that is repeated 30 times, can readily be commented out, leaving just enough displayed geometry to serve as reference and to provide the needed set of vertices for the next set of facets to be added, while providing good visibility to the inner parts.

The next phase focuses on the construction of the innermost shell of the complete 2-manifold. For this phase, much of the outer framework may be turned off; however, it is advisable to keep visible the area where the prototypical lobe mesh was edited, so that the new inner mesh portions are constructed directly below it, and those regions can eventually be joined and connected to one another (Fig.8c).



(a) (b) (c)

Figure 8: *Construction of a 3-level Star Cinder: (a) 30 outer lobes, (b) laterally connected into the outermost shell; (c) one prototypical radial stem connecting to the innermost shell.*



(a) (b)

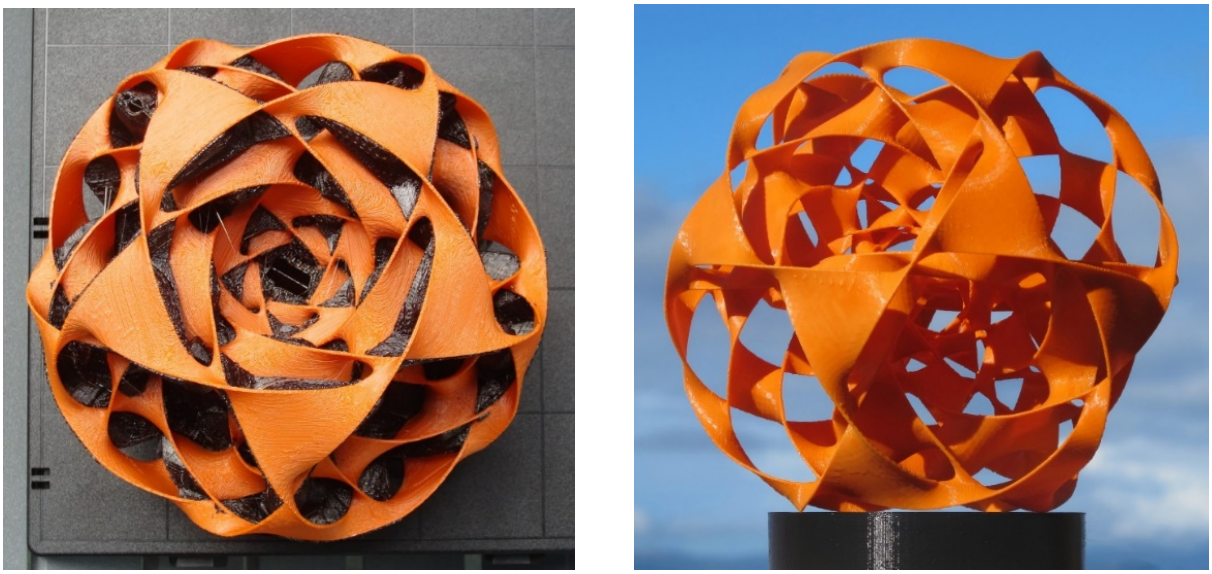
Figure 9: *3-level Star Cinder: (a) complete CAD model; (b) 3D ABS print (18cm diameter).*

Figure 9a shows the complete CAD model with all three nested shells of lateral connections. Figure 9b shows a 3D print made in ABS plastic on a *Dimension 1200* machine from Stratasys [18], after some of the parameters that define the radius and the tilt of the 30 border circles have been slightly changed so as to

produce narrower outer flanges and a somewhat enlarged center section. The internal support structure needed during layered fabrication consisted of wax (P400SR), which then was removed cleanly by immersing the 3D print in a concentrated sodium hydroxide solution for several hours.

This was actually a rather difficult design. Near the center of the sphere, the dense tangle of the 30 border circles made it difficult to figure out how to place the representative quadrilaterals and what border curves they should connect. About eight trial iterations were needed: placing some tentative facets; then letting the procedural specification instantiate them with the full icosahedral symmetry; and then removing them again if they resulted in some inacceptable intersections. The process stretched through 3 or 4 session over a couple of days; it also involved making slight changes to the border circles, hoping that this would make it easier to see where connecting facets should be placed.

The problem is that there is no unique solution to creating a fully symmetrical 2-manifold in such a tangle of border curves. A slight change in the tilt angle of the 30 circular border curves with respect to the corresponding dodecahedron edges will readily accommodate a topologically quite different soap-film surface. The result shown in Figure 10 now has five nested shells, but of somewhat simpler geometry. This model was also fabricated on a *Dimension 1200* fused-deposition-modeling machine from Stratasys, using ABS plastic with soluble wax support.

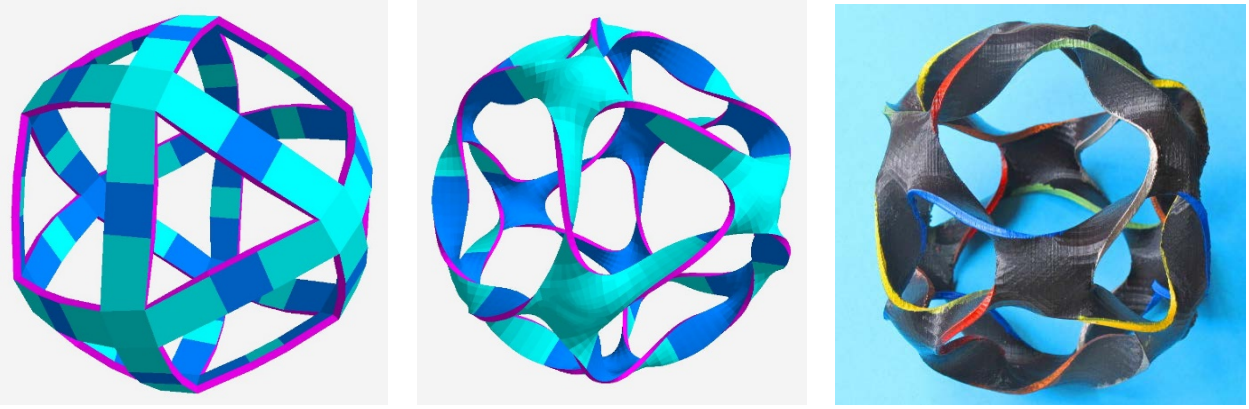


(a) (b)

Figure 10: *5-level Star Cinder: (a) 3D ABS print coming off the printer; (b) with support removed.*

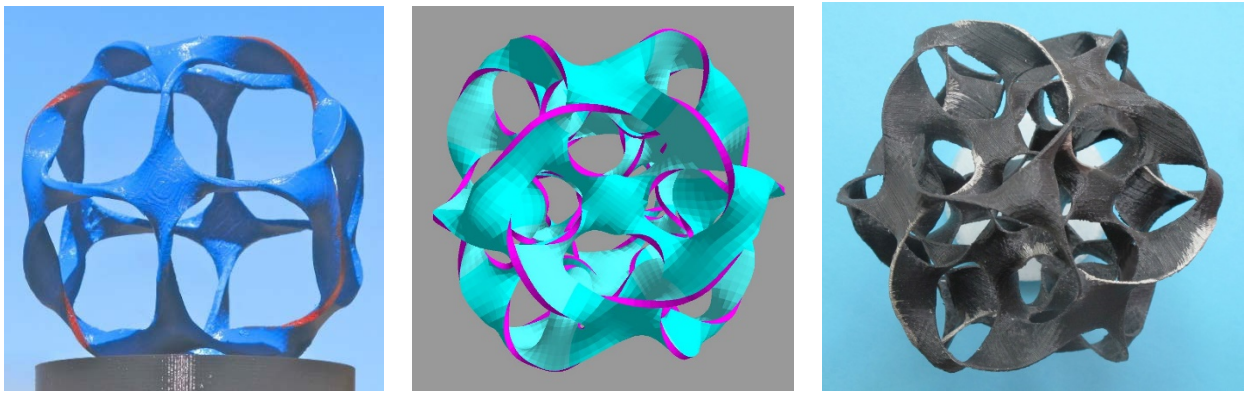
5. A Different Approach to Making *Star-Cinder* Geometries

Because of the difficulties with placing appropriate surface elements that result in a proper 2-manifold with no self-intersections and which exhibits the full symmetry of the oriented icosahedron, we also looked for other ways to design such multi-shell geometries. An “obverse” approach starts with a nested framework of twisted ribbons following the edges of a semi-regular polyhedron and then tries to interconnect concentric shells with additional ribbon elements running in the radial direction. These radial connections are particularly easy to define between two nested shells that are geometrical duals of one another. As one example, we started with a ribbon structure following a cuboctahedron (Fig.11a). All the ribbons connecting neighboring vertices were twisted through 180° (Fig.11b). In this approach, the topology of the surface is preconceived; on the other hand the resulting structure of the border curves is far from clear. Even on the simple geometry of the cuboctahedron frame, it was difficult to visualize the shape of individual border loops. Clarity was gained by physically painting different border loops in different colors. On the twisted cuboctahedron frame, there are 6 border loops with 4-fold symmetry (Fig.11c; see yellow color).



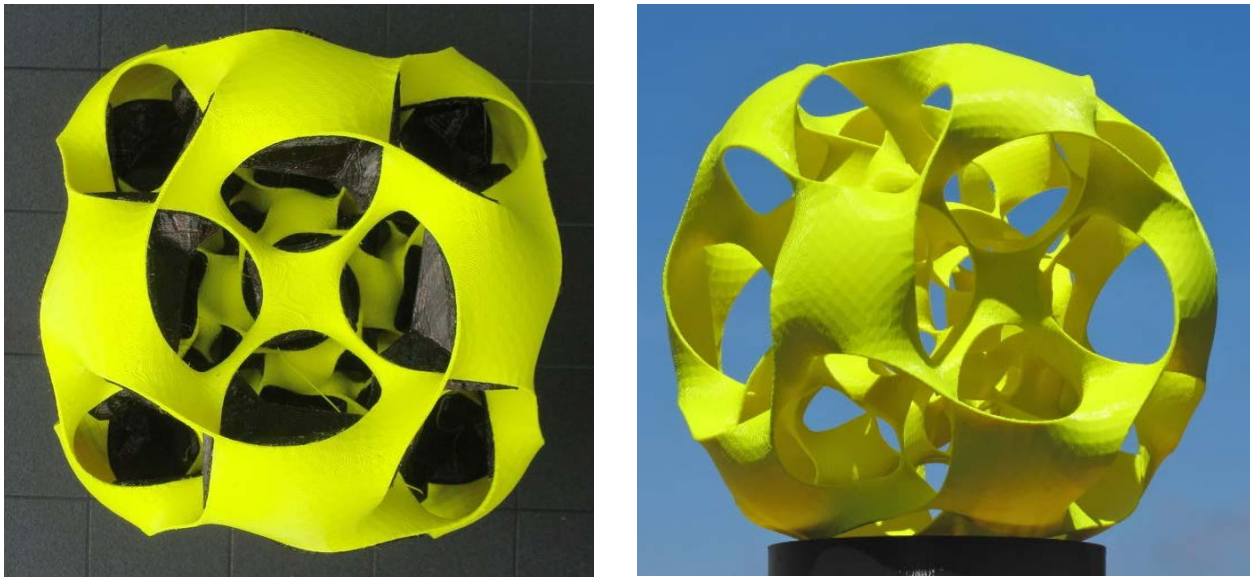
(a) (b) (c)

Figure 11: *Cuboctahedron frame: (a) with flat ribbons; (b) with twisted ribbons; (c) 3D print.*



(a) (b) (c)

Figure 12: *(a) Rhombic dodecahedron frame; (b) combined CAD model; (c) 3D print of 2-level surface.*



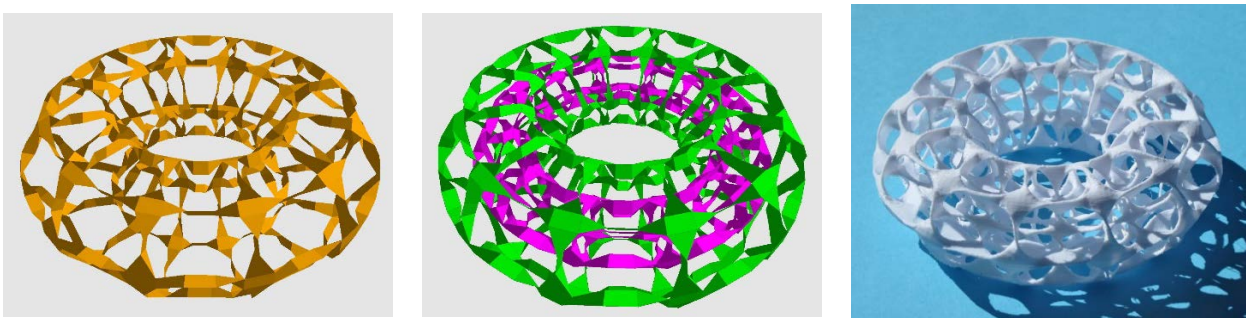
(a) (b)

Figure 13: *3 shells of twisted ribbon frames: (a) as it comes off the printer; (b) with support removed.*

The dual structure is the rhombic dodecahedron. On its twisted frame structure, there are also 6 border loops with 4-fold symmetry (Fig.12a). Next, a scaled-down version of it is placed inside the cuboctahedron frame. At the lined-up edge midpoints, both dual ribbons crossing at right angles have a radially oriented tangent. Thus, it is easy to connect them with a short ribbon segment that twists through 90° (Fig.12b). On the resulting combined 2-shell surface (Figs.12c), the border loops have 3-fold rotational symmetry.

With this approach, it is easy to make star cinders with three or more levels of nested shells. Once the twisted ribbon frames have been generated for a pair of dual polyhedra, more multi-shell surfaces can readily be constructed by nesting scaled version of the two shells in an alternating sequence. Once the two levels of connecting radial tabs have also been defined, those quadrilaterals can also readily be replicated with simple uniform scaling. Figure 13 shows a 3-level structure with a cuboctahedron frame on the outside. This model was also fabricated with soluble support. Figure 13a shows how it emerges from the printer, with the dark wax support still in place; Figure 13b shows the final sculpture after the support structure has been washed away.

This approach is not limited to shells of genus zero. In Figures 14, the same technique has been applied to a torus surface. The surface is partitioned into an array of 12 6 quadrilateral facets, and the edges in this wire frame are replaced with twisted ribbons (Fig.14a). A skinnier torus with the same major radius is placed inside, with the tessellation of its surface shifted by half a facet size in both directions of its surface coordinate system. Thus again, the two frames can be seen as duals of one another with corresponding edges crossing at right angles (Fig.14b). At these crossing points, where the set of ribbons on the outer, as well as on the inner torus, have surface tangents that point in the radial direction, connecting flanges between the outer and the inner frames are introduced; they are just single quads that twist through 90° as they sweep in the radial direction. In this case, two sets of six quads had to be introduced in the graphical display, oriented in the longitudinal direction as well as in the latitudinal direction, respectively. Both sets follow one minor circular loop in the torus. The set of those 12 facets is then put into the group that replicates one such minor-loop assembly twelve times around the *z*-axis (Fig.14c). With this approach, we knew exactly what had to be done, and the whole process took only about five hours, spread over two sessions in two consecutive days.



(a) (b) (c)

Figure 14: *Twisted torus frames: (a) outer torus; (b) concentric tori; (c) 3D print of 2-level surface.*

6. NOME Development

After a couple of somewhat ad-hoc implementations of NOME, the front-end of the current NOME system is structured like a compiler. A parser scans the textual input and constructs an abstract syntax tree (AST), whose nodes are the statements and expressions. From this information, NOME builds the directed, acyclic scene graph (DASG). During the rendering process, the DASG is expanded into the fully instantiated scene tree. All nodes in the DASG have pointers back to the AST to remember where they came from. Through these pointers, any edits that a user makes interactively can be traced back to the input NOME file and can be recorded with minimal deviations from what the user wrote originally.

Update Efficiency

Typically, our hierarchical descriptions comprise a few hundred to a few thousand vertices. Today's (lap-top) computers are fast enough to rebuild the whole scene from scratch within a fraction of a second. Thus, a continuous slider change will appear as a smooth deformation of the overall assembly. However, after the *merging* operation and a few levels of subdivision, the instantiated scene tree is much larger, and it may contain hundreds of thousands of vertices. Even small parameter changes may dislocate previously coinciding edges, and thus prevent a merger of border edge segments. For consistency, the merge operation has to be redone after the hierarchical geometry description has been updated. Merging, subdivision, and offsetting are costly operations, and here it does pay to determine exactly what portions of the scene geometry must be reconstructed.

In the current NOME, a dependency graph co-exists with the DASG and shares many of the same nodes. Every object in the dependency graph may have one or more inputs and outputs. For example, a *point* object has an output that tells its downstream connections about the *position* and the *name* of the point. In order to maintain an interactive frame rate, we separate the update process into two phases: *marking dirty* and the *actual update*. Every object output has a flag that denotes dirtiness, indicating the need to update it before its values can be used. Whenever an output is marked dirty, the dirty flag propagates downwards to all inputs that it is connected to. Each of those object inputs then decides, based on a predefined function per input, whether to propagate the dirty flag further down the dependency graph. As an example: Any user interaction with a slider will mark its *value* output dirty. A *point* object, that happens to use said value, in turn marks its *position* output dirty, but its *name* need not be marked dirty. The dirty flags then propagate to all faces and meshes that use that point, until, eventually, they reach the scene tree nodes associated with the meshes. During the second phase, the scene tree is traversed again, and any object that previously had been marked dirty is now recalculated from the latest clean nodes in the graph.

During a scene update, certain objects may become deleted. For example, when the user lowers the segment count for a cylinder, many strips of vertices and faces will simply disappear, invalidating all constructs that are dependent on them. Clearly, the program should gracefully notify the user of this fact – and not just crash! In addition to the dirty flag, we also maintain a *validity* flag that tracks the status of the most recent update for each object. The *validity* flag propagates downwards during the update process, and an object will delay its update actions as long as it sees that one of its upstream objects is still invalid.

Deleted Geometry

One of the issues that gave us a headache when trying to re-integrate graphical edits into the original procedural NOME file was deleting some geometry. Adding new geometry does not seem to cause any problems; it can simply be added at the end of the original NOME file in as compact a form as possible. The designer can then move these additional NOME statements to a suitable location in the hierarchical description, so that the new elements become part of an overall symmetrical replication. However, deleting geometry is a different matter!

The designer may realize at some point that the procedural symmetry operation places some faces on top of one another and that some of the prototype faces that were introduced earlier may have to be removed. The graphical operations of selecting such faces and marking them for removal, can easily achieve this. The situation is different, if a designer wants to cut an opening into a torus that was produced by a procedural generator routine. If it is just an isolated facet, the situation is as above: the face can be deleted, and all vertices stay in place. The question arises, whether vertices that are no longer needed to support any face, when an extended collection of faces are being deleted, should remain in place or should also be deleted from the data structure. Our current view is that those vertices stay in place and that the “deleted” faces are simply marked *invisible*. This has the advantage that any other constructs that may make use of some of these vertices remain unaffected.

The same question becomes more critical, if some geometry higher up in the hierarchy is being deleted. What if a whole boundary curve is deleted? Will all facets attached to it become obsolete? To minimize

this problem, NOME has no operation in its graphical interface to remove anything except faces. But a corresponding problem still exist when a user changes the number of sampling points along a curve via an interactive slider. If vertex #49 no longer exists, because the number of sampling points has been changed from 50 to only 25, constructs referring to vertex #49 will become ill-defined. The designer will then have to choose a different anchor point. In the current implementation, any geometry that relies on vertex #49 is temporarily marked *invalid* and not displayed until the said vertex somehow shows up later.

7. Potential NOME Enhancements

It may seem rather tedious for the designer to have to refer back to the NOME file in order to integrate some newly added facets into the overall hierarchical structure, or to introduce a new parameterized vertex to provide additional support for a large connecting membrane between multiple border curve segments. During the development of NOME, we have contemplated ways of integrating such geometry-changing operations into the graphical editing domain. Mostly we have concluded that this would be rather challenging, and, in the end, still fairly limiting. Any additional editing capabilities would require a corresponding set of menu options, and the user could then do only what these menus offer to do. Letting the designer go back to the text-based description offers a flexible and less constrained way of enhancing a given design. However, some possible enhancements are currently being discussed and may be implemented in the near future:

Adding Localized Spline Sampling Points:

In the vicinity of a close, skewed crossing of two border curves, where the surface will have to produce a twisted saddle, it is desirable to have sampling points on both border curves near the closest approach of the two curves. Appropriate new sample points could be introduced on demand by the designer. However, there are some conceptual difficulties with naming these vertices and with maintaining full symmetry in the overall design. So far, we have found it easier to just increase the overall sampling density on the border curve and then let the designer use only a subset of these vertices to define new facets and bridging surfaces.

Smoothing of Border Curves:

In the merged, hierarchically-flat scene description, a sequence of border vertices in a coarse polyhedral description of a 2-manifold could be redefined as the control vertices of a (cubic) B-spline, and this new border curve could then be sampled with the same number of points as there were original "control" vertices. The old border vertices are then replaced with the corresponding new sample points, which now form a smoother border curve. All surface elements are automatically re-attached to the new border curve. This smoothing process could be repeated.

Generating Smoother Surfaces

The final surface geometry depends on the way in which the designer picks facets to define the surface topology. It is desirable to reduce this dependency. There are a few techniques through which this goal can be achieved in the future. In 1992 Henry Moreton described a system for fairing an arbitrary polyhedral surface by computing tangency constraints at all the vertices and then replacing all edges with smooth curve segments that all meet at the vertices with G1-(tangent)-continuity. All original facets get replaced by more complex curved patches [12].

Ken Brakke has developed the *Surface Evolver* [3] to approximate very closely true minimal surfaces. It would be nice to have this capability integrated in NOME and use it in place of the much simpler Catmull-Clark subdivision [4]. This integration will not be trivial, and its use will require additional user interaction. Brakke's surface refinement is not totally automatic; the user will have to use some judgement to initiate alternate refinement steps such as: *equiangulation*, *vertex averaging*, or *jiggling*. Also, the border curves need to be parameterized in a form that is compatible with Brakke's surface refinement process.

8. Discussion and Conclusions

To make possible the design of the complex, multilevel *Star-Cinder* sculptures, we needed an environment that combines parameterized, procedural specification of border curves with an interactive graphical way to define the surface topology suspended on those curves. A key challenge was to maintain an easy-to-understand, high-level description of the current state of the sculpture, which included the graphical edits made through the last interactive session.

The real bottleneck impeding the speed of some of the more complex surfaces remains at the conceptual design level. The times spent in clicking on vertices to add new facets, or switching back and forth between the graphical display and the NOME text file, or fixing up this text file to be in its most succinct form with a clean hierarchical structure, – they all pale in comparison with the thinking time spent observing the current display, rotating it, zooming in and out, in order to figure out the right place to insert the next facet. Thus, while adding additional, higher-level, graphical editing capabilities may be nice, they will not have a big impact on speeding up overall design time.

The development of NOME has stretched over the last three years [19],[6]. It started out as some special "scaffolding" to realize the design of some complex geometrical 2-manifolds; but then it raised some broader conceptual issues about combining procedural specifications with graphical editing capabilities. At this time, the NOME system has been used almost exclusively by the authors of this paper, The presented options may not be the best final solution, but they constitute a reasonable, practical approach for designing some rather complex 2-manifolds. The current development state is presented here to open up a broader discussion on how the integration between two quite different domains may best be achieved.

References

[1] Autodesk. 3DS MAX – https://www.autodesk.com/products/3ds-max/overview

[2] Blender. "Home of the Blender Project." – https://www.blender.org/

[3] K. Brakke. "Surface Evolver." Experimental Mathematics, 1 (2): pp 141–165 (1992) – https://en.wikipedia.org/wiki/Surface_Evolver

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

[5] R. Chugh, B. Hempel, M. Spradlin, J. Albers. "Programmatic and Direct Manipulation." SIGPLAN 2016 – https://arxiv.org/abs/1507.02988

[6] G. Dieppedalle. "Interactive CAD Software for the Design of Free-form 2-Manifold Surfaces." Tech Report (EECS-2018-48), 5/10, 2018. – https://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-48.html

[7] N. Gabo. "Translucent Variation on Spheric Theme." – https://www.guggenheim.org/artwork/1381

[8] B. Hempel and R. Chugh. "Semi-Automated SVG Programming via Direct Manipulation." UIST 2016 – https://arxiv.org/abs/1608.02829

[9] E. Hild. "EVA HILD (Home Page)." – http://evahild.com/

[10] A. Holden. "Orderly Tangles." Columbia University Press, New York, 1983.

[11] Maya. "Computer Animation & Modeling Software." – https://www.autodesk.com/products/maya/overview

[12] H. P. Moreton and C. H. Séquin. "Functional Optimization for Fair Surface Design." Computer Graphics 26, 2, July 1992. – https://people.eecs.berkeley.edu/~sequin/CS284/TEXT/p167-moreton.pdf

[13] C. O. Perry. "Dual Universe." – http://www.charlesperry.com/sculpture/dual-universe

[14] P. Perry. "Selected Works 1964-2011." Page 168. The Perry Sudio, Norwalk, CT (2011).

[15] Rhino3D. "Rhinoceros." – https://www.rhino3d.com/

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

[17] C. H. Séquin. "Disk(5,3)" – http://people.eecs.berkeley.edu/~sequin/TALKS/2019/Disk_5_3_CODE.txt

[18] Stratasys. "Dimension 1200es." – http://usglobalimages.stratasys.com/Main/Files/Machine_Spec_Sheets/PSS_FDM_Dim1200es.pdf

[19] Y. Wang. "Robust Geometry Kernel and UI for Handling Non-orientable 2-Mainfolds." UCB Tech Report (EECS-2016-65), 5/12, 2016. – https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-65.html