Developing fractal curves

Geoffrey Irving* Henry Segerman†

Version 1, October 31, 2012

Abstract

Many fractal curves can be produced as the limit of a sequence of polygonal curves, where the curves are generated via an iterative process, for example an L-system. One can visualise such a sequences of curves as an animation that steps through the sequence. A small part of the curve at one step of the iteration is close to a corresponding part of the curve at the previous step, and so it is natural to add frames to our animation that continuously interpolate between the curves of the iteration. We introduce sculptural forms based on replacing the time dimension of such an animation with a space dimension, producing a surface. The distances between the steps of the sequence are scaled exponentially, so that self-similarity of the curves is reflected in self-similarity of the surface. For surfaces based on the constructions of certain fractal curves, the approximating polygonal curves self-intersect, which means that the resulting surface would also self-intersect. To fix this we smooth the polygonal curves. We outline two very different approaches to producing and smoothing the geometry of such a "developing fractal curve" surface, one based on a direct parameterisation, and the other based on Loop subdivision of a coarse polygonal control mesh.

1 Fractal Curves

Space filling and other kinds of fractal curves have been a popular subject for many mathematical artists. See for example Helaman Ferguson [3] and Carlo Séquin [16]. Often a fractal curve is generated as the limit of a sequence of polygonal curves $(c_1, c_2, \ldots)$, where each step c_n is obtained from its predecessor c_{n-1} via some local process. By *locality* of the process, we mean that a small part of c_n is close to, and determined by only a small part of c_{n-1}. For example, for the terdragon curve (Davis-Knuth [2], see Figure 1), each edge of c_{n-1} is replaced by three edges in c_n close to the edge of c_{n-1}, and the positions and orientations of those three edges depend only on the position and orientation of the edge of c_{n-1}. One way to formalise this notion of locality is to say that the curve can be generated via an *L-system*.

L-systems are named after Aristid Lindenmayer, the biologist who invented the concept as a way to describe elementary plant development [11]. An L-system is a string rewriting system. We have an alphabet of symbols V, a string of symbols ω defining the start of the system, or *axiom*, and a set of production rules P determining how we produce a new string from an old string. The *L-system grammar* is the collection of these three parts, $G = (V, \omega, P)$. This generates a sequence of strings by repeatedly applying the rules P, starting with the axiom ω.

*Email: irving@naml.us, Otherlab, San Francisco, CA, United States

†Email: segerman@unimelb.edu.au, Department of Mathematics and Statistics, University of Melbourne, VIC 3010, Australia

Example 1.1. *If we take the following L-system grammar*

$$G_{terdragon} = (\{F, -, +\}, F, \{F \mapsto F-F+F\})$$

then the output of the L-system is the following sequence of strings.

$$\begin{array}{ccccccccccccccccccccccccccc}
 & & & & & & & & & & F & & & & & & & & & & & & & & \\
 & & & & F & & & & - & & & & F & & & & + & & & & F & & & & \\
F & - & F & + & F & - & F & - & F & + & F & + & F & - & F & + & F & - & F & + & F & - & F & + & F \\
 & & & & & & & & & & \vdots & & & & & & & & & & & & & &
\end{array}$$

Note that not all symbols need to have a replacement rule.

Given such an L-system, we may view the produced strings as instructions for building a curve in a "turtle graphics" fashion [15]. That is, we start somewhere on the plane, with a specified "forward facing" direction. We read the input string from left to right, and interpret the symbols as actions to perform. In the above example, we make the following interpretations:

F	move forward one unit
$-$	turn right 120°
$+$	turn left 120°

The resulting sequence of curves are the steps in the construction of the *terdragon curve*, shown in Figure 1.

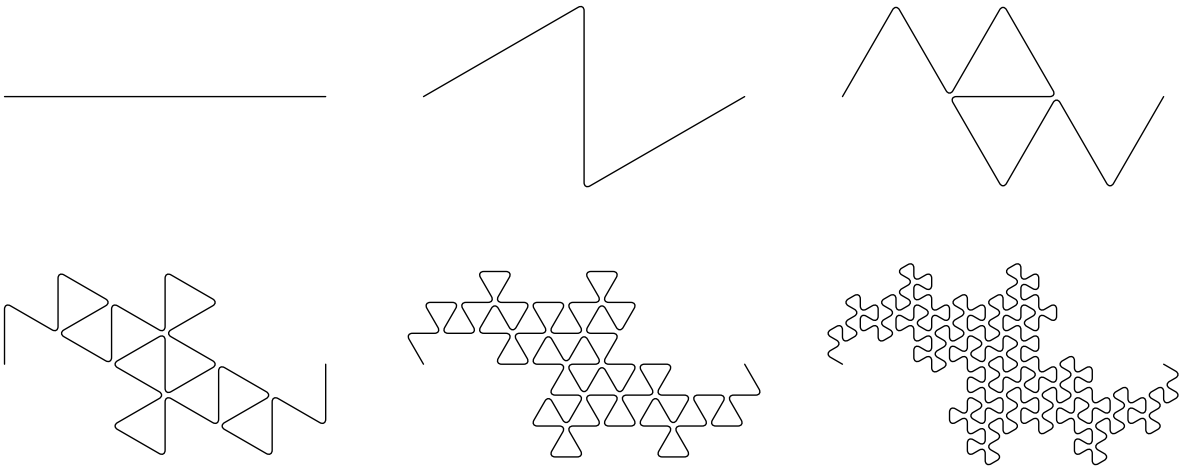


Figure 1: Six steps in the construction of the terdragon curve.

There are a few things to note about these curves. First we have chosen the starting direction and the step size for our turtles so that they all end at the same position relative to the starting position. With these modifications, we can see that each line segment is replaced by three segments in a zig-zag pattern at the next step. Second, we have rounded the corners slightly. If we did not do this then later versions of the curve would self-intersect. It would be more faithful to the actual curve to draw it with the self-intersections, but this way we can see that it is indeed a curve (rather than a graph), and also see the order in which it traverses the various edges. However, an unfortunate feature of the curves in Figure 1 is that the radius of the rounding of the corners is fixed, rather than depending on the lengths of the line segments

of the curve. This makes the curves at different steps locally less similar to each other than they could be. We can fix this, as in Figure 2, by having the rounding radius depend on the lengths of the original line segments. Here, we have chosen the rounding so that the path is a union of arcs of circles, other than at the endpoints.

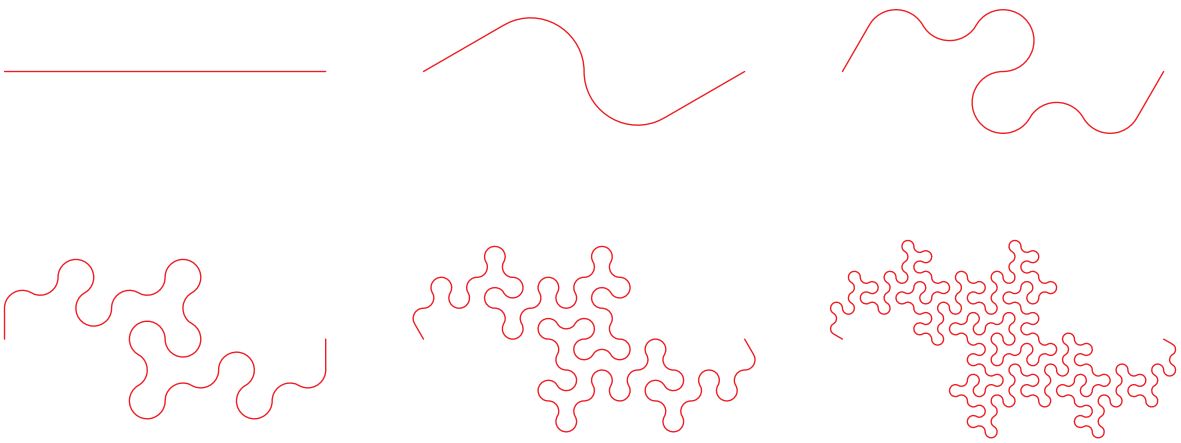


Figure 2: Six steps in the construction of the terdragon curve, with rounding at the corners based on the lengths of the line segments at each step.

We note a couple of numerical features of this process: As mentioned before, each line segment is replaced by three line segments, and the lengths of the line segments at each new stage are scaled down by $1/\sqrt{3}$ from the previous stage.

Another beautiful example is the *(Heighway) dragon curve* [4, 2], with L-system

$$G_{\text{dragon}} = (\{F, X, Y, -, +\}, FX, \{X \mapsto X-YF, Y \mapsto FX+Y\}).$$

In this case, F is again interpreted as a move forwards, $-$ and $+$ are turns by $90°$, while X and Y are ignored when we convert to a curve. This time one line segment is replaced by two, each of which is scaled down by a factor of $1/\sqrt{2}$. See Figures 3 and 4.

2 Time $\leftrightarrow$ Space

Given the locality of the modification process, it is natural to think about interpolating between the steps to make a continuous process. One could continuously morph between the different steps of the construction. This has been done by Jeffrey Ventrella, and animations of this for the dragon, terdragon and many other fractal curves are available on YouTube. To our knowledge however, nobody has yet considered replacing the time dimension with a space dimension, converting the sequence of curves in $\mathbb{R}^2$ into a surface in $\mathbb{R}^3$.

In Ventrella's animations, each successive step of the construction is animated in an equal period of time. While this makes sense in the 2-dimensional animated setting, we found that there is a better choice for surfaces. In particular, if we choose the vertical separation carefully, then the resulting surface retains the self-similarity properties that the curves do. See Figure 5.

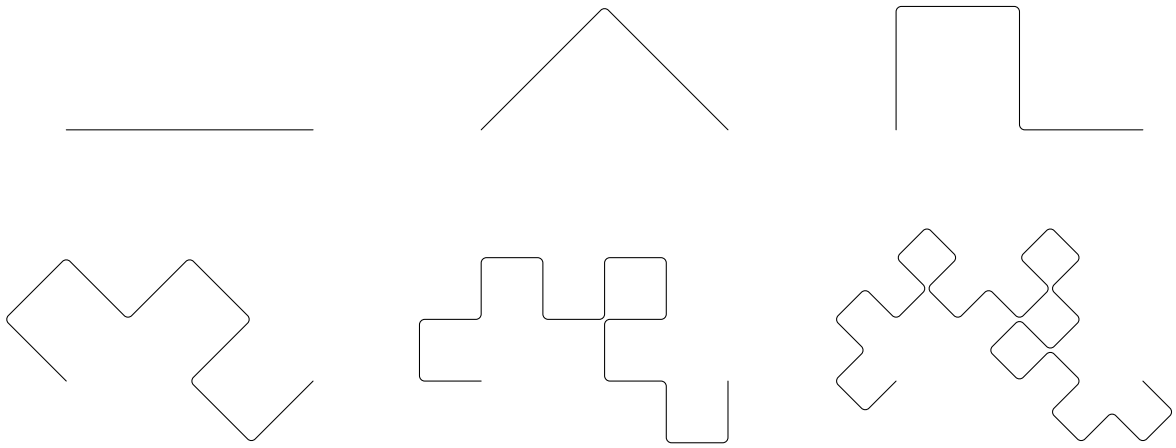


Figure 3: Six steps in the construction of the dragon curve.

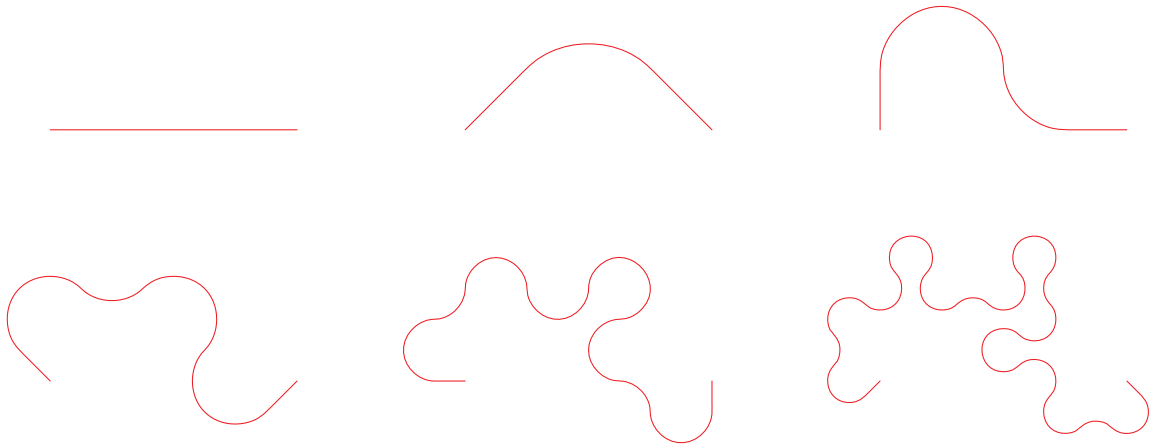


Figure 4: Six steps in the construction of the dragon curve, with rounding at the corners based on the lengths of the line segments at each step.

This shows the first three steps in the dragon curve construction (with no rounding), connected to each other in the simplest way possible, by linear interpolation. If the surface is to exhibit the same self-similarity that the curves do, then we must have that $l_1/h_1 = l_2/h_2$. Since we know for this sequence of curves that $l_2 = l_1/\sqrt{2}$, then we get that $h_2 = h_1/\sqrt{2}$.[1] In general, to ensure self-similarity in the surface, the vertical separation between two adjacent curves should be scaled to be proportional to the lengths of the line segments in, say, the lower curve. (Note that the lengths of line segments in the upper curves are a constant multiple of those in the lower, so we could equally say that the vertical separations should be proportional to the line segment lengths in the upper curve!)

[1] As an interesting consequence, an interpolating surface for the dragon curve sequence can be made using isosceles right triangles and equilateral triangles, as shown in the figure.

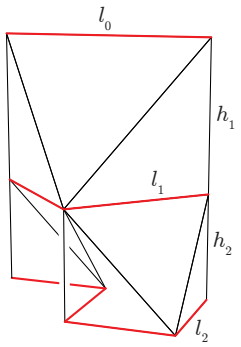


Figure 5: The first three steps of the dragon curve construction, joined by a linearly interpolating surface. The vertical separation between the curves is chosen so that the surface is self-similar.

3 Implementation

Now that we have a rough idea of the surfaces we want to build, we have some choices in exactly how to define the surface, and how to turn it into a 3-dimensional volume either for 3D printing or rendering. In particular, we have to deal with thickening up a 2-dimensional surface so that it has nonzero volume. One could (correctly) argue that the above discussion on similarity also applies to the thickness of the thickened surface. That is, if our sculpture is to retain the self-similarity properties of the sequence of curves, then the thickness should also scale in the same way. See Figure 6 for steps of the terdragon curve sequence, thickened in a self-similar way.

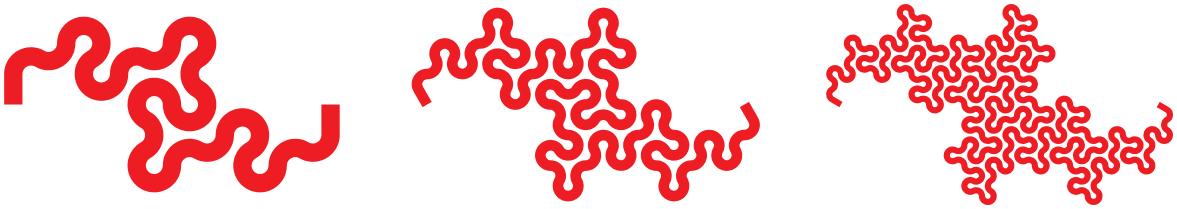


Figure 6: Three of the steps in the construction of the terdragon curve, with line thickness proportional to line segment length.

However, there are some downsides to doing things this way in the case of a physical sculpture. Depending on the material used, the thinnest objects that can be safely 3D printed are a little under a millimetre thick. This puts a lower bound on how thin the lowest level of the surface can be, and so puts a lower bound on how thin the highest level can be, assuming the self-similarity condition. In our case, a developing surface representation for the terdragon curve showing 6 levels would have the top being $\sqrt{3}^{6-1} \approx 15.6$ times thicker than the bottom. This would be significantly over-engineered in terms of strength, and would also be more expensive. For these reasons, we decided to 'cheat' on the thickness for 3D printing, inviting the viewer to interpret the sculpture as a 2-dimensional surface, with the thickness of the object a necessary evil only. However, our computer renders follow the "correct" thickness scaling.

As discussed above, for some fractal curves the steps of the construction self-intersect. Since our surface interpolates between the curves, it will also self-intersect unless we smooth it in some way. In principle, one could smooth only in the horizontal slices, and use linear interpolation so connect the curves to each other vertically. However, it seems better to smooth in both directions from an aesthetic point of view.

Given all of the above requirements, the two authors of this paper approached the problem of designing a sculpture of a developing fractal curve in very different ways. In the next two sections, we outline our strategies.

3.1 Direct parameterisation

H.S.'s solution will perhaps be more familiar to many of our readers, so we discuss it first. With this strategy, we build a function of three variables $F\colon [0,1] \times [0,1] \times [-\epsilon, \epsilon] \to \mathbb{R}^3$, mapping a thin cuboid with coordinates (s,t,r) into $\mathbb{R}^3$. The first coordinate, s moves us between different curves in the construction of the fractal curve, the second coordinate t moves us along each curve (and interpolated curves between them), and the third coordinate r corresponds to the thickened direction. We could abstractly define a function involving infinitely many steps of the construction, so that the line $\{(1,t,0) \mid t \in [0,1]\}$ maps to the fractal curve itself. However, for making an actual physical sculpture, we will cut the process off at some point.

3.1.1 Smooth curves

For the jth step of the construction, we are given a polygonal curve $p_i\colon [0,n] \to \mathbb{R}^3$, where p_i is linear on $[j, j+1], j \in \mathbb{Z}$, and constant height depending on i. We use Bézier curves to smooth these out. A cubic Bézier curve, evaluated over an interval, can interpolate between start and end points with specified tangent vectors at the start and end. See Figure 7. For the start and end points for each Bézier curve, we take the midpoints of the line segments of p_i, together with the start of the first edge and the end of the last. That is, our points are $\{p_i(t) \mid t = 0, \frac{1}{2}, \frac{3}{2}, \ldots, \frac{2n-3}{2}, \frac{2n-1}{2}, n\}$. We choose the magnitudes of the tangent vectors so that the curves approximate arcs of circles. After scaling the input appropriately, this gives us piecewise polynomial functions $c_i\colon [0,1] \to \mathbb{R}^3$.

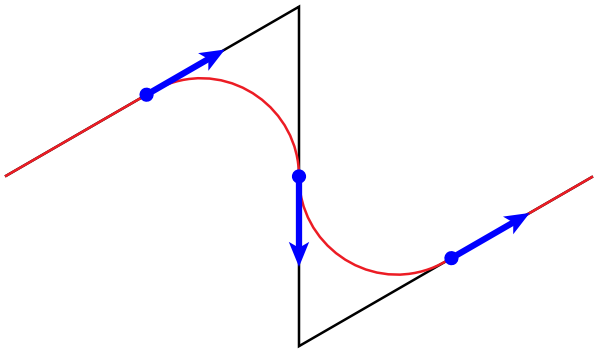


Figure 7: Smoothing a polygonal curve using cubic Bézier curves.

Next, we need to interpolate between the different curves. We can use exactly the same techniques for this as well! Before we constructed polynomials in t, with coefficients being

vectors in $\mathbb{R}^3$ that interpolated between points in $\mathbb{R}^3$. Now we construct a polynomial in a new variable s, with coefficients being polynomials in t, which interpolates between our polynomials c_i. The endpoints for the ith Bézier polynomial are the vector polynomials c_i and c_{i+1}. For the tangent polynomial at the ith level, we use $(c_{i+1}(t) - c_i(t))/2$, apart from at the first and last levels, where we use a constant polynomial whose value is a downwards pointing vector. Again, we scale the input so that s ranges between 0 and 1.

3.1.2 Thicken to a function of 3 variables

We now have a parametric function of two variables, say $f\colon [0,1] \times [0,1] \to \mathbb{R}^3$, where $f(s,t)$ gives a point on our desired surface. We can then convert this into our function F of three variables as follows. First we define a normal function using the vector cross product:

$$n(s,t) := \frac{\partial f}{\partial s}(s,t) \times \frac{\partial f}{\partial t}(s,t)$$

Then we define F by

$$F(s,t,r) := f(s,t) + n(s,t)r.$$

This would give a constant thickness surface. In fact, this choice would also run into problems of structural strength of the 3D printed object. The thickness of the surface would have to be relatively thin to avoid collision with other parts of the surface near the last level. Then constant thickness would force us to have large thin sheets near the first level, which would be relatively fragile. To fix this problem, we used a decreasing function of s for r that interpolates between a large enough thickness to safely print large sheets for s near 0, and a small enough thickness to avoid collisions for s near 1. In fact the function we chose is also exponential, but with a slower rate of decrease than that given by similarity.

3.1.3 Build the surface

We are now in a position to build the geometry that becomes the data sent to a 3D printer to produce the sculpture. We used NURBS surfaces (which can be thought of as generalisations of Bézier curves) to define the boundary of the object. These can be thought of as parameterised surface patches. In principle, one could take each of the six rectangular faces of the cuboid $[0,1] \times [0,1] \times [-\epsilon, \epsilon]$ and map each forwards using the function F to produce six parameterised surface patches that define the boundary of our sculpture. However, there are some technical difficulties in following this strategy. In particular, the data specifying a NURBS surface patch gives an approximately equal density of information about the geometry at different parts of the patch. In the case of $F(s,t,\epsilon)$ however, the surface becomes far more complex (exponentially so) as s increases. For this reason, we need to break up the rectangles in the s and t directions, and use more NURBS surface patches for larger s.

We also want to introduce a pattern of holes in the surface. There are a number of benefits to doing this. First, punching holes in the surface reduces the total volume of the object, and so the cost in production. Second, it allows us to see partially through the surface to other features behind. Third, it allows us to highlight features of the parameterisation of the surface. In our case, it is natural to punch holes in a way that respects and highlights the self-similarity of the surface. For the surface generated from the dragon curve, the natural repeating "tile" for the object is the five sided polygon made from three triangles in Figure 5.

See also Figure 8(a). We punch a hole in each of these tiles. Figure 8(b) shows a subdivision of a neighbourhood of the edges of the tile into tapered quadrilaterals, which is consistent under the similarity transforms to the other tiles. When we thicken this pattern up in the normal direction, we get a pattern of tapered cuboids, the boundary faces of which give our parameterised surface patches.

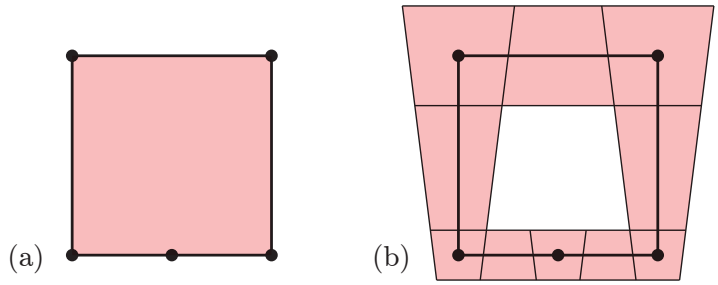


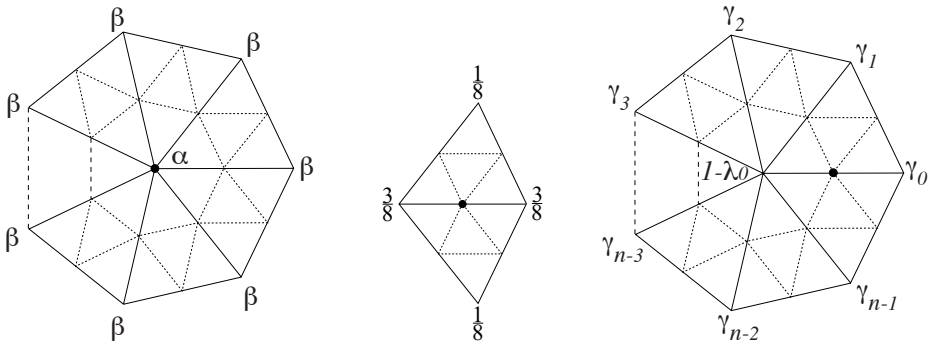
Figure 8: In (a), the repeating tile for the developing dragon curve. In (b), we subdivide a neighbourhood of the edges of the tile into tapered quadrilaterals.

See Figure 14 for the resulting sculpture. For sculptures based on the other fractal curves, we vary the pattern of tapered quadrilaterals. For the terdragon curve for example, the bottom edge of the tile is divided into three segments rather than two, and we introduce an extra horizontal bar halfway up the tile. We add the extra horizontal bar because that the terdragon curve has greater changes between iterations, in comparison with the dragon curve (primarily because one line segment becomes three, rather than the two line segments for the dragon curve). This means that the surface would be broken up too much if there were only a single large hole in each tile. The extra horizontal bar maintains the visual continuity of the surface. To produce designs based on curves with an even larger number of edges replacing a single edge, we would further subdivide the respective tiles.

3.2 Loop subdivision

G.I. chose a more general route to smooth surfaces using the technique of Loop subdivision [12] with the modifications in [13]. We start with the original unsmoothed polygonal curves at each level (Figure 3 without the rounded corners), shift each internal vertex inwards a small amount to avoid self intersections, and join the curves at consecutive levels into a minimal triangular mesh as shown in Figure 5. The result is a triangle mesh homeomorphic to $[0,1]^2$, which we will denote T_0. Starting with this coarse *control mesh*, Loop subdivision defines a sequence of subdivided meshes $\{T_n\}$ converging towards a smooth limit surface.

Before giving the details in the surface case, we describe the analogous process for a one dimensional curve. Say we have a doubly infinite sequence of points $\{p_{0,i}\}_{i\in\mathbb{Z}}$ defining a piecewise linear *control curve* γ_0 at level 0. To define a refined curve γ_{n+1} from γ_n, we introduce an additional vertex between each pair of vertices of γ_n. The most general symmetric way to



$$\tau = \frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{d} \qquad u_j = \cos\frac{2\pi j}{d}$$

$$\alpha = \begin{cases} 1 - \frac{\tau(4+\tau(5\tau-8))}{2(1-\tau)} + \tau^2, & d \geq 6 \\ \frac{81-5\sqrt{5}}{128}, & d = 5 \\ \frac{1}{2}, & d = 4 \\ \frac{7}{16}, & d = 3 \end{cases} \qquad \gamma_j = \begin{cases} \frac{2\tau^3}{d(1-\tau)}(1+u_j)\left(\frac{1}{\tau} - \frac{3}{2} + u_j\right)^2, & d \geq 6 \\ \frac{3+\sqrt{5}}{32}\left(\frac{5-\sqrt{5}}{5} + u_j\right)^2, & d = 5 \\ \frac{1}{2}\left(\frac{1}{2} + \frac{3}{8}u_j\right)^2, & d = 4 \\ \frac{1}{6}\left(\frac{5}{4} + u_j\right), & d = 3 \end{cases}$$

$$\beta = \frac{1-\alpha}{d} \qquad \lambda_0 = \sum_j \gamma_j$$

Figure 9: Loop subdivision stencils for vertices of degree d (left), ordinary edges adjacent to two degree 6 vertices (middle), and extraordinary edges incident to a degree d vertex (right). If an edge touches two non-degree 6 vertices, the right stencil is applied in both directions and averaged. Each stencil defines the position of the large dot indicating a vertex in the subdivided mesh indicated as a convex combination of vertices from the mesh from one level up. Reproduced with permission from [13].

define the positions of the new vertices as averages of immediate neighbours is

$$p_{n+1,2i} = \alpha p_{n,i} + \frac{1-\alpha}{2}\left(p_{n,i-1} + p_{n,i+1}\right)$$
$$p_{n+1,2i+1} = \frac{1}{2}\left(p_{n,i} + p_{n,i+1}\right)$$

where $\alpha \in [0,1]$. That is, a new vertex is positioned at the average position of the two vertices it comes between, while the old vertices are moved in a way depending on their positions and the positions of their neighbours. The question is how much weight to give to the old position of a vertex, versus the old positions of its neighbours. The weight α can be decided on by considering the behavior of the subdivision process around one of the vertices, say $p_{n,0}$. The nth neighbourhood $(p_{n,-1}, p_{n,0}, p_{n,1})$ is a linear function of the $(n-1)$th neighbourhood, and eigenanalysis of the resulting operator can be used to determine the local smoothness of the curve. It turns out that the limit curve is C^2 continuous if and only if $\alpha = 3/4$, though we omit the details. In fact, the $\alpha = 3/4$ case reduces to cubic B-spline interpolation [10].

In the surface case, T_n is refined to T_{n+1} by introducing a new vertex along each edge, and splitting each triangle into four child triangles. Each vertex position in T_{n+1} is defined as a convex combination of its local neighbourhood in T_n. Vertices on the boundary of a mesh use the curve subdivision rules given above if smoothness is desired, or are left at their T_n

positions to indicate a sharp corner (we used no sharp corners in our examples). The weights for interior vertices are shown in the *stencils* in Figure 9. Each vertex in T_n is moved to a new position in T_{n+1}, which is a weighted average between its position and the average of its neighbours, as in the left diagram of Figure 9. Each remaining vertex in T_{n+1} corresponds to an edge in T_n. If both endpoints of the edge are of degree 6 then the position is computed as in the middle diagram of Figure 9, using only the positions of the vertices of the two triangles adjacent to the edge in T_n. Otherwise (that is, if either endpoint is an *extraordinary* vertex of degree $\neq 6$), the right diagram of Figure 9 is used, involving all vertices adjacent to each extraordinary vertex.

See Figure 10 for the first levels of Loop subdivision on the polygonal surface given by the dragon curve construction.

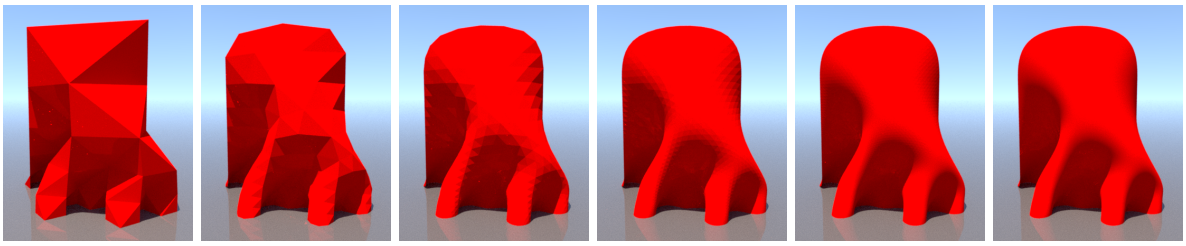


Figure 10: Levels 0 through 5 of Loop subdivision for a simple dragon surface. The final levels are indistinguishable from the limit surface.

Away from an *extraordinary* vertex of degree $\neq 6$, the limit surface of the Loop subdivision process reduces to cubic B-splines. At an extraordinary vertex, the surface degrades to C^1 with bounded but discontinuous curvature. As in the curve case, the weight formulas above can be derived via eigenanalysis of the linear neighbourhood subdivision operator around a vertex to ensure bounded curvature and non-negative weights [13].

Unfortunately for curvature continuity, extraordinary vertices with degree ≥ 6 abound in our control meshes: for the dragon curve (as in Figure 5), internal vertices alternate between degree 6 and 7 (see Figure 13(a)). The surface we wish to build is combinatorially equivalent to a square, and of course it is possible to triangulate a square so that all (interior) vertices have degree at most 6. However, this ignores the self-similarity of the desired geometric surface. From a combinatorial point of view, all of our surfaces can be seen as being made up of fundamental tiles as in Figure 8(a). That is, we have a square tile, where the bottom edge is subdivided n times, where n depends on the iterative process that generates the fractal curve. We call this an *n-way branching tile*. In the two examples we have seen so far, the dragon curve has $n = 2$, while the terdragon curve has $n = 3$. The left side of the tile is glued to the right side of another tile and vice versa, while each of the segments at the bottom of the tile is glued to the top side of another tile. Thus, the tile fills the half plane $\mathbb{R} \times (0, \infty)$ as in Figure 11 (for $n = 2$). A *self-similar* triangulation of the surface is a triangulation which subdivides the tiling of the surface, so that the induced triangulations of the tiles are combinatorially identical.

So, our control mesh is a self-similar triangulation (which is clearly a natural choice), but it has many extraordinary vertices. Due to poor results in the past with similar meshes, we were

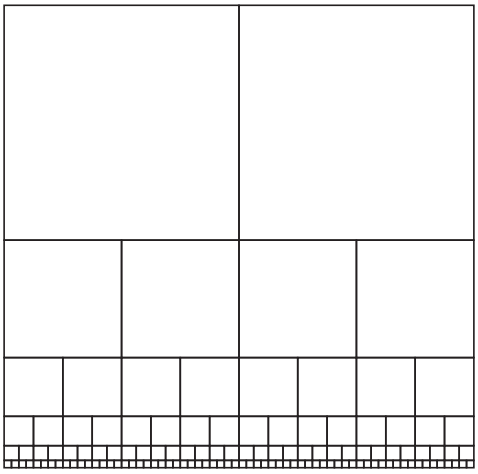


Figure 11: The fundamental tile with 2-way branching covers the half plane $\mathbb{R} \times (0, \infty)$.

initially concerned that Loop subdivision would fail to produce aesthetically pleasing surfaces. It is fortunate that this fear proved unfounded, because a self similar triangulation with all vertices of degree 6 is impossible for our surfaces, as we will see in Corollary 3.2. To show this, we will produce a nonnegatively curved metric on any purely degree 6 self-similar triangulation where the area of geodesic balls grows exponentially, which contradicts the following simple theorem (see e.g. Theorem 107 of [1] for a tightened, generalized version):

Proposition 3.1. *Let M be a Riemannian surface which is flat except at a discrete set of positively curved cone points with surrounding angle $< 2\pi$ (a so-called* Euclidean cone metric*). Then the area of a geodesic ball of radius r around any point is at most πr^2.*

Proof. Let p be a flat point of M with tangent space T_p, $B_r(p)$ the ball of radius r around $p \in M$, and $D_r(0)$ the ball (actually a Euclidean disk) of radius r around $0 \in T_p$. Let $S_r \subset D_r(0)$ be the set of vectors for which the exponential map $\exp : T_p \to M$ is uniquely defined. That is, S_r consists of all $v \in T_p$ for which $\exp\big([0,1)v\big)$, the geodesic from p moving in the direction of v, up to but not including the point at distance r, consists of only flat points. We will show that $\exp(S_r) = B_r(p)$. If not, choose $q \in B_r(p) \setminus \exp(S_r)$ and let $\gamma : [0,1] \to M$ be a minimum length path from p to q. Since $q \notin \exp(S_r)$, there must be some $0 < t < 1$ such that $\gamma(t)$ is a cone point; pick the first such t. Since the total angle around $\gamma(t)$ is strictly less than 2π, the angle on one of the two sides of γ must be strictly less than π. But then we can shorten γ by moving it slightly to that side as shown in Figure 12, contradicting the minimality of γ. This shows that $\exp(S_r) = B_r(p)$. Since exp is metric preserving on S_r, we have

$$\text{area}(B_r) = \text{area}(\exp(S_r)) \leq \text{area}(S_r) = \pi r^2$$

The same inequality holds if p is a cone point by taking limits. □

Corollary 3.2. *Suppose that we have a self-similar triangulation of the half plane tiled with n-way branching tiles. Then there exists a vertex with degree > 6.*

In fact, because of the self-similarity, there must be at least one extraordinary vertex per tile.

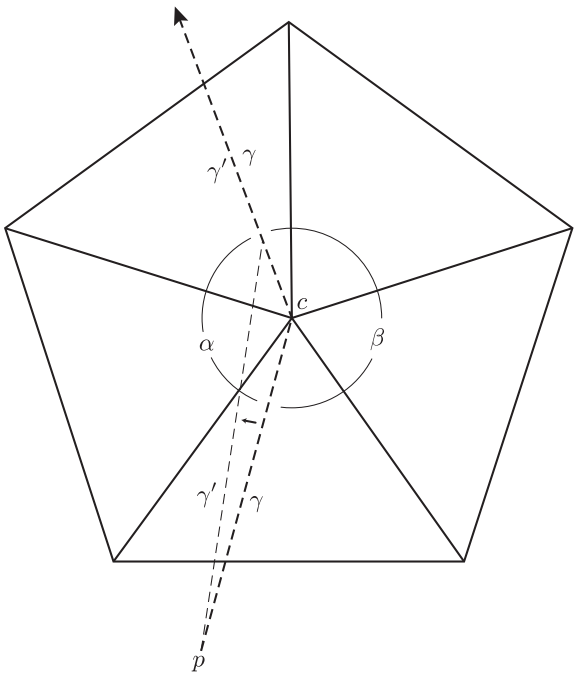


Figure 12: Consider a piecewise geodesic curve γ in a surface which is flat except at positively curved cone points such as c. Since c is positively curved, its surrounding angle satisfies $\alpha + \beta < 2\pi$. Thus one of the two sides has angle $< \pi$, say $\alpha < \pi$, and the length can be reduced by shifted slightly towards this side to produce γ'.

Proof. Suppose M is any manifold triangulation with vertex degrees ≤ 6. Following [7], we give M a Euclidean cone metric by making each triangle equilateral with unit length sides. This metric is flat except at vertices of degree $d < 6$, which are positively curved with total surrounding angle $2\pi d/6 < 2\pi$. In the case of a branching self-similar triangulation, the diameter of each tile is constant, so the volume of a ball grows asymptotically exponentially since it will encompass exponentially many tiles (see Figure 11). This contradicts Proposition 3.1, so at least one vertex of degree > 6 must exist. □

We originally searched for a combinatorial proof of Proposition 3.2 in terms of "average vertex degree", but failed due to the counterexample shown in Figure 13(b). Indeed, since the boundary of a region in a self-similar tiling is an asymptotically constant fraction of the size of the interior, it is unclear whether or not the concept of average vertex degree can be well defined.

As in the case of explicit parameterization, the final step in our construction is to thicken the surface into a region of nonzero volume. To do this in the Loop subdivision case, we define a scalar thickness field on the vertices of the control mesh, scaling exponentially as a function of level. In the case of rendered images there is no physical minimum thickness constraint, and we are free to use the "correct" exponential scaling to ensure self-similarity of the 3D volume. Since the stencil weights in the Loop subdivision process are a function of topology only, we can apply these same weights to the scalar thickness field to define a sequence of refined scalar fields on the vertices of each refined mesh, converging towards a smooth limit field. We then

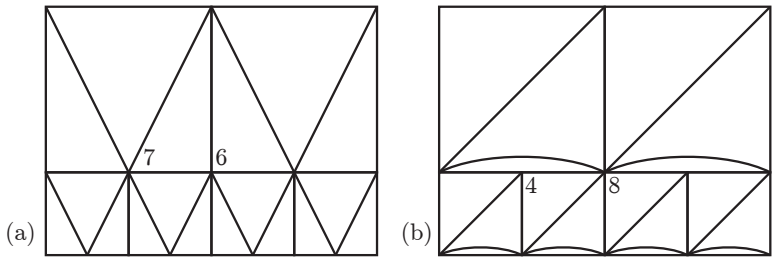


Figure 13: The "average vertex degree" of a self-similar triangulation of a 2-way branching tiling tiling is not constant: (a) has average degree 6.5 while (b) has average degree 6.

offset along the normal direction by half the thickness to each side, and connect boundaries to produce a closed mesh.

4 Results

We applied our techniques to the constructions of a number of fractal curves. The associated L-system grammars are listed in Table 1, together with the turning angle associated to the $-$ and $+$ symbols. For completeness, we list the dragon and terdragon curves again here. We give the L-system refinement level in each figure caption, counting the L-system axiom as level 0.

Table 1: L-system grammars.

Name	Alphabet	Axiom	Production rules	Angle
Dragon curve	$\{F, X, Y, -, +\}$	FX	$\{X \mapsto X-YF, Y \mapsto FX+Y\}$	$\pi/2$
Terdragon curve	$\{F, -, +\}$	F	$\{F \mapsto F-F+F\}$	$2\pi/3$
Hilbert curve	$\{F, X, Y, -, +\}$	$+YF-XFX-FY+$	$\{X \mapsto +YF-XFX-FY+,$ $Y \mapsto -XF+YFY+FX-\}$	$\pi/2$
Sierpinski arrowhead curve	$\{F, X, Y, -, +\}$	XF	$\{X \mapsto YF-XF-Y,$ $Y \mapsto XF+YF+X\}$	$\pi/3$
Koch snowflake	$\{F, -, +\}$	$F++F++F$	$\{F \mapsto F-F++F-F\}$	$\pi/3$
Gosper curve	$\{F, X, Y, -, +\}$	FX	$\{X \mapsto X+YF++YF-FX-$ $-FXFX-YF+,$ $Y \mapsto -FX+YFYF++YF+FX-$ $-FX-Y\}$	$\pi/3$

4.1 Direct parameterisation

In Figures 14 and 15 we show photographs of the resulting 3D printed sculptures for the dragon curve and the terdragon curve. We also applied our techniques to the Hilbert curve [6] and the Sierpinski arrowhead curve [14], see Figures 16 and 17.

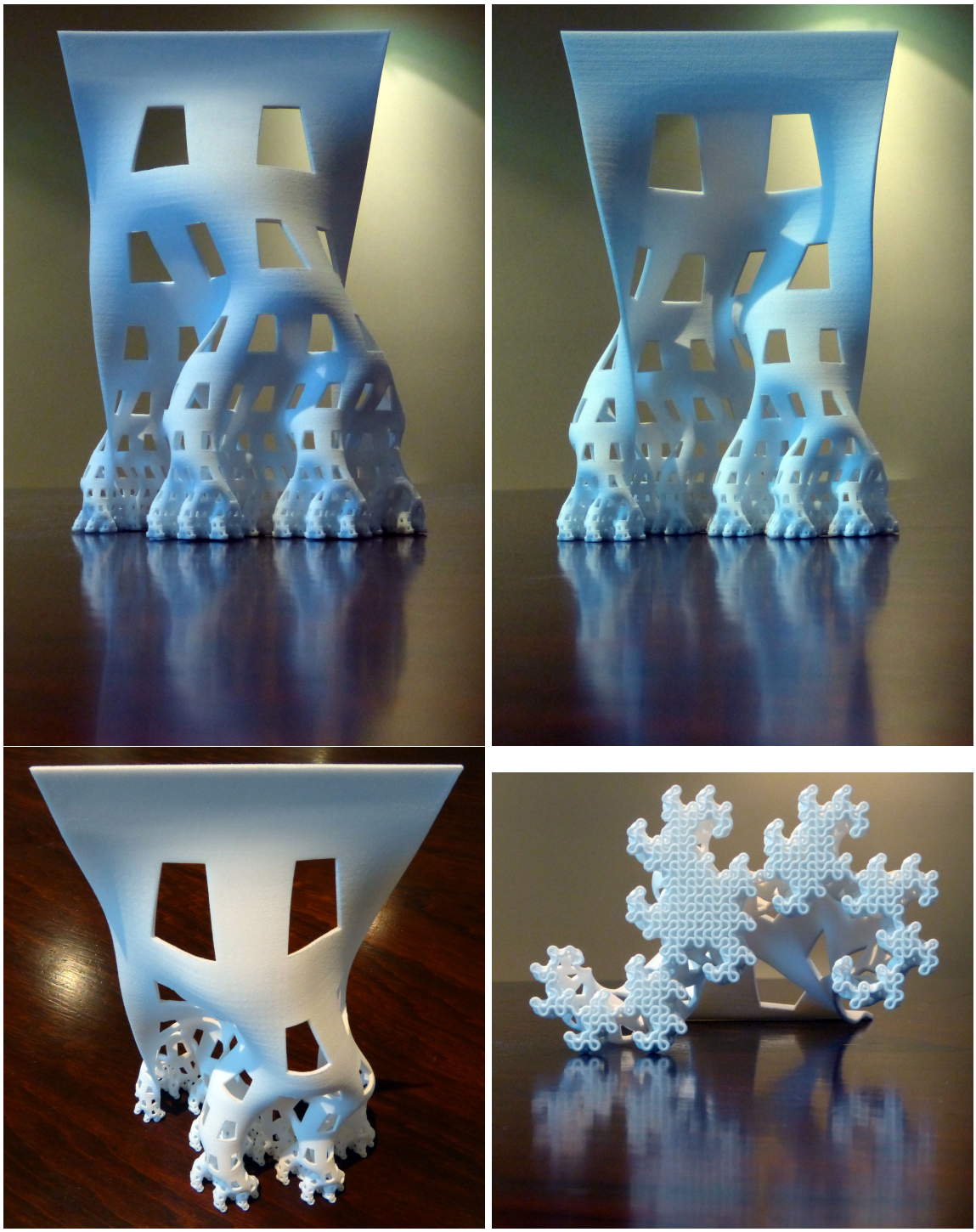


Figure 14: The developing dragon curve up to level 10 (constructed by direct parameterisation) as a 3D print.

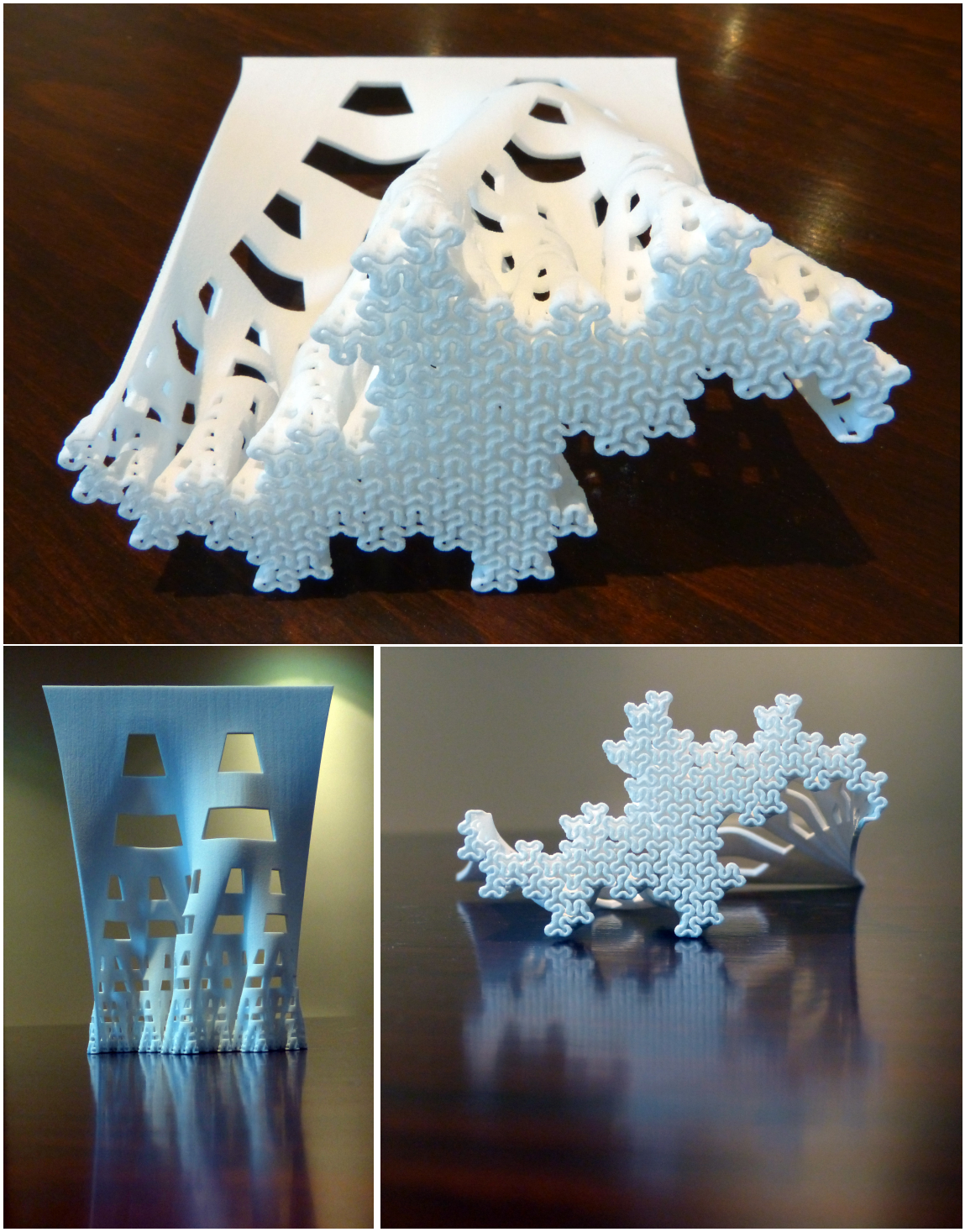


Figure 15: The developing terdragon curve up to level 6 (constructed by direct parameterisation) as a 3D print.

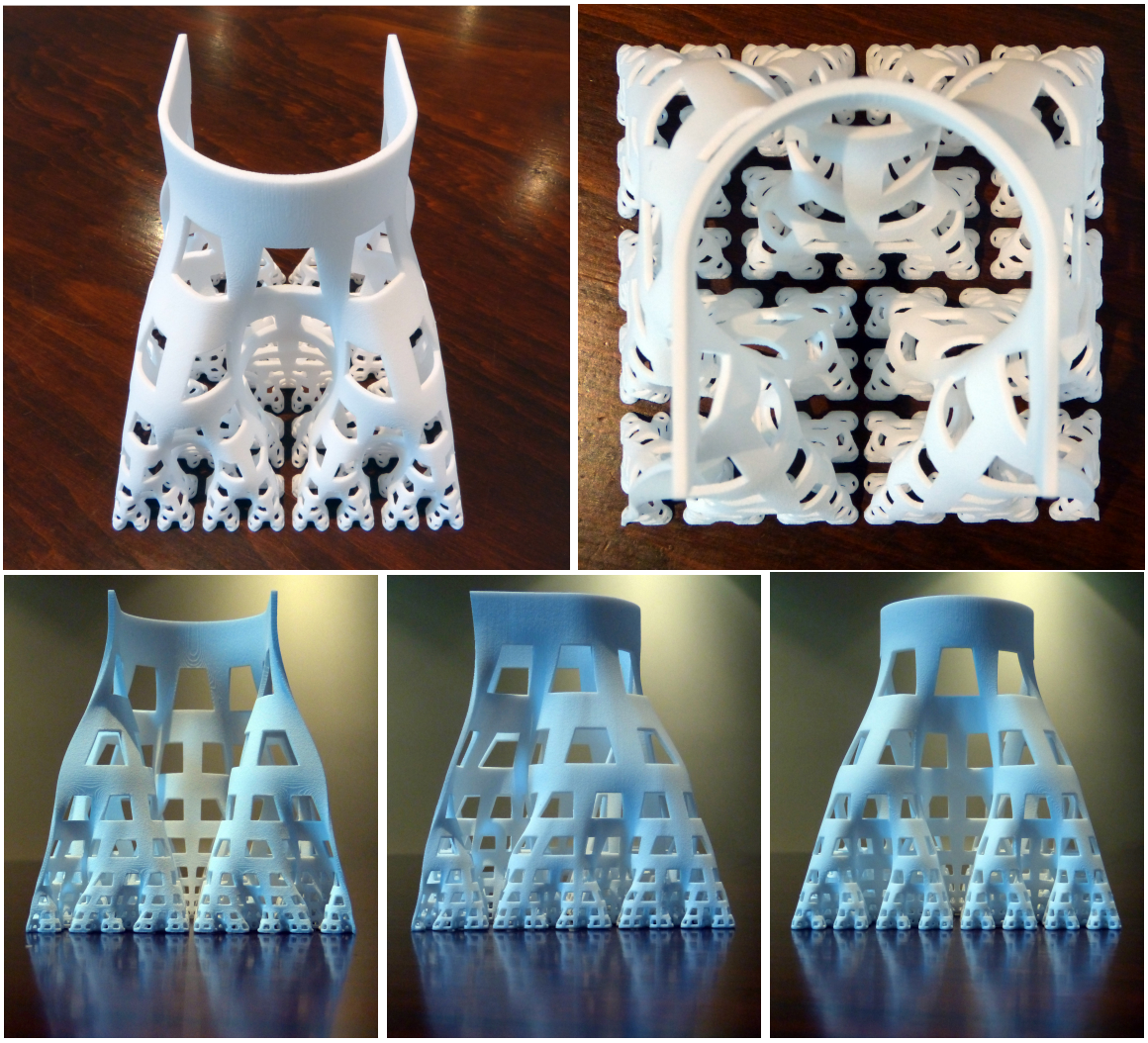


Figure 16: The developing Hilbert curve up to level 4 (constructed by direct parameterisation) as a 3D print.

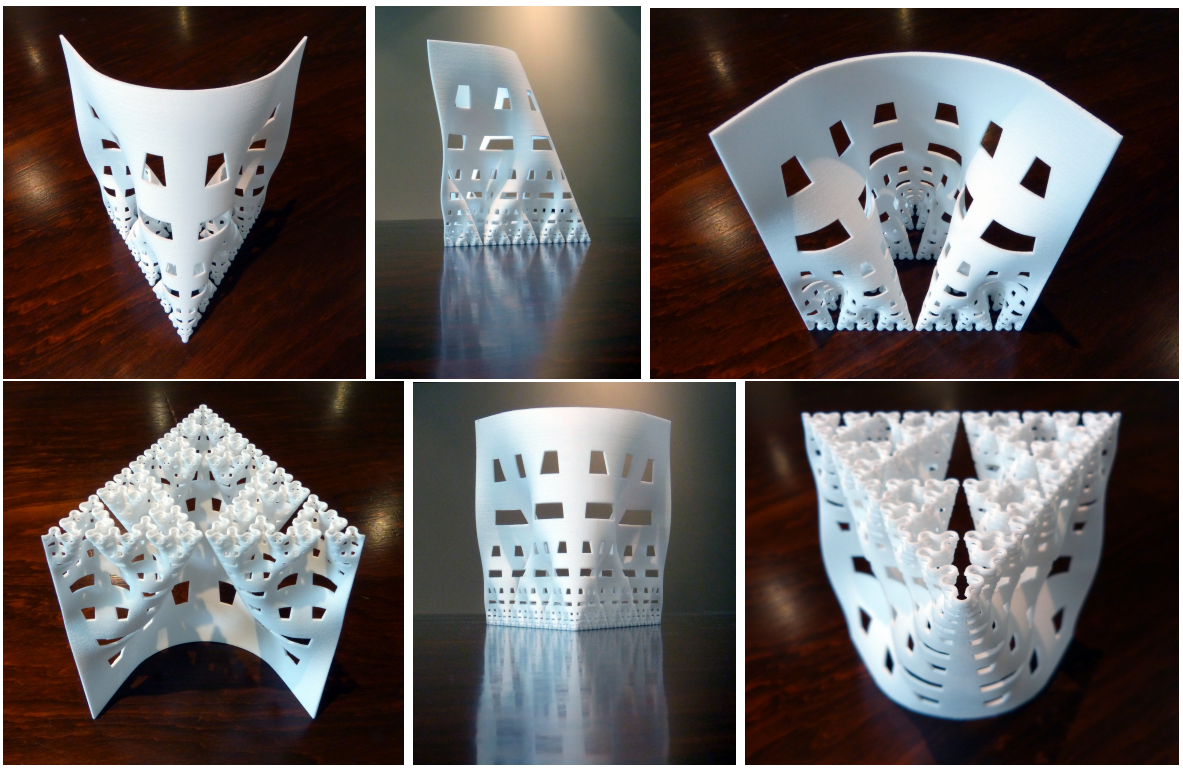


Figure 17: The developing Sierpinski arrowhead curve up to level 6 (constructed by direct parameterisation) as a 3D print.

4.2 Loop subdivision

Renderings of Loop subdivided fractal surfaces for the dragon and terdragon curves are shown in Figures 18, 19, and 20. We also rendered developing versions of the Koch snowflake [9] and Peano-Gosper curve [5], see Figures 21 and 22. These images were generated using the Mitsuba physically based renderer [8] with guidance from Eugene d'Eon.

The combinatorial structure of our surfaces is self-similar in the sense that each tile in the self-similar triangulation is triangulated in the same way. However, we can ask if combinatorial similarity is reflected in geometric similarity of the generated surface. By construction, in level 0 of the Loop subdivision all tiles are similar to each other. When we start smoothing however, this can change. The tiles of the renders are collected into equivalence classes under the similarity relation, and all tiles in each equivalence class given the same colour. The number of these equivalence classes grows with refinement level for the first few levels then reaches a constant due to self-similarity. Since the shape of a Loop subdivided tile depends on vertices up to two steps away in the control mesh as well as proximity to the boundary, these numbers are fairly large: 131 for the dragon curve, 135 for the terdragon curve, 125 for the Koch snowflake, and 1487 for the Gosper curve.

The code for the Loop subdivided variant of developing fractal curves is available at `https://github.com/otherlab/fractal` under a BSD license.

References

[1] Marcel Berger, *A panormaic view of Riemannian geometry*, Springer Verlag (2003).

[2] C. Davis and D. E. Knuth, *Number Representations and Dragon Curves*, Journal of Recreational Mathematics, **3** (1970), 66–81 and 133–149.

[3] Helaman Ferguson, *Two Theorems, Two Sculptures, Two Posters*, American Mathematical Monthly, **97** (1990), no. 7, 589–610.

[4] Martin Gardner, *Mathematical Games: An Array of Problems that Can Be Solved with Elementary Mathematical Techniques*, Scientific American, March 1967, 124–129.

[5] ———, *Mathematical Games: In which "monster" curves force redefinition of the word "curve"*, Scientific American, **235** (1976), 124–133.

[6] David Hilbert, *Ueber die stetige Abbildung einer Linie auf ein Flächenstück*, Mathematische Annalen **38** (1891), no. 3, 459–460.

[7] Ivan Izmestiev, Robert B. Kusner, Günter Rote, Boris Springborn, and John M. Sullivan, *There is no triangulation of the torus with vertex degrees 5, 6, . . . , 6, 7 and related results: Geometric proofs for combinatorial theorems*, arXiv:1207.3605 (2012).

[8] Wenzel Jakob, *Mitsuba renderer*, `http://www.mitsuba-renderer.org` (2010).

[9] Helge von Koch, *Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire*, Arkiv för Matematik, Astronomi och Fysi, **1** (1904), 678–704.

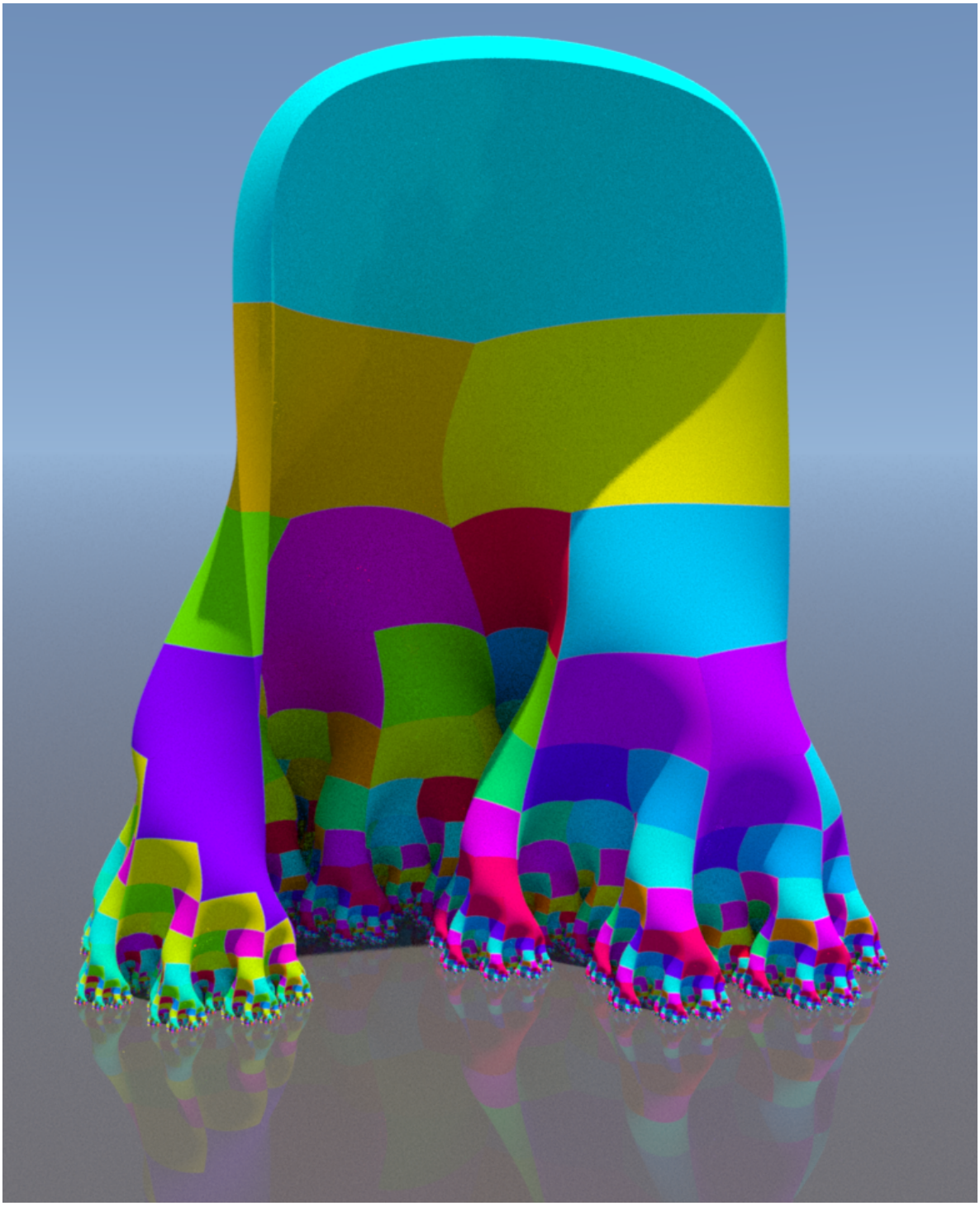


Figure 18: A render of the developing dragon curve up to level 17 (constructed by Loop subdivision).

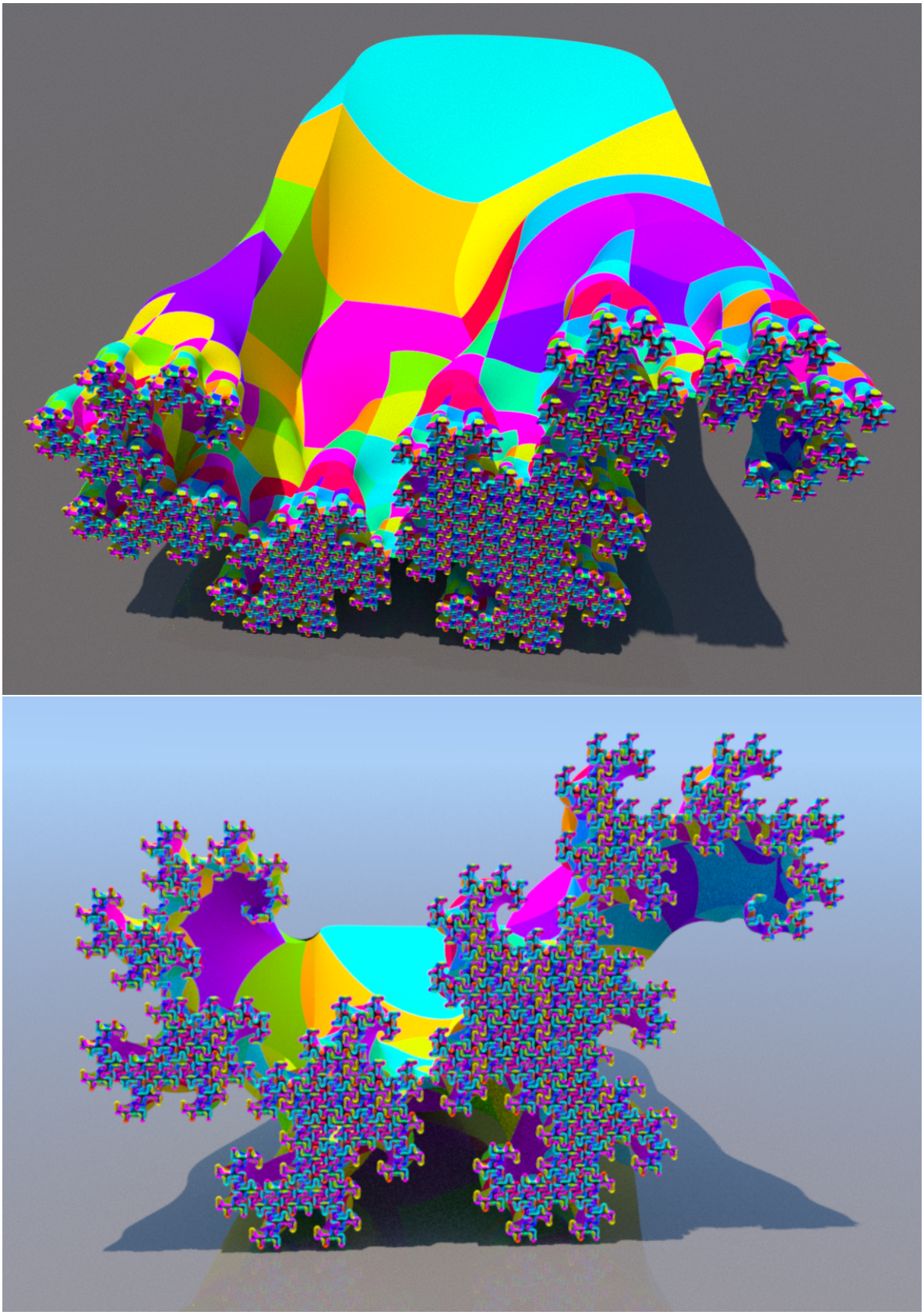


Figure 19: Renders of the developing dragon curve up to level 13 (constructed by Loop subdivision).

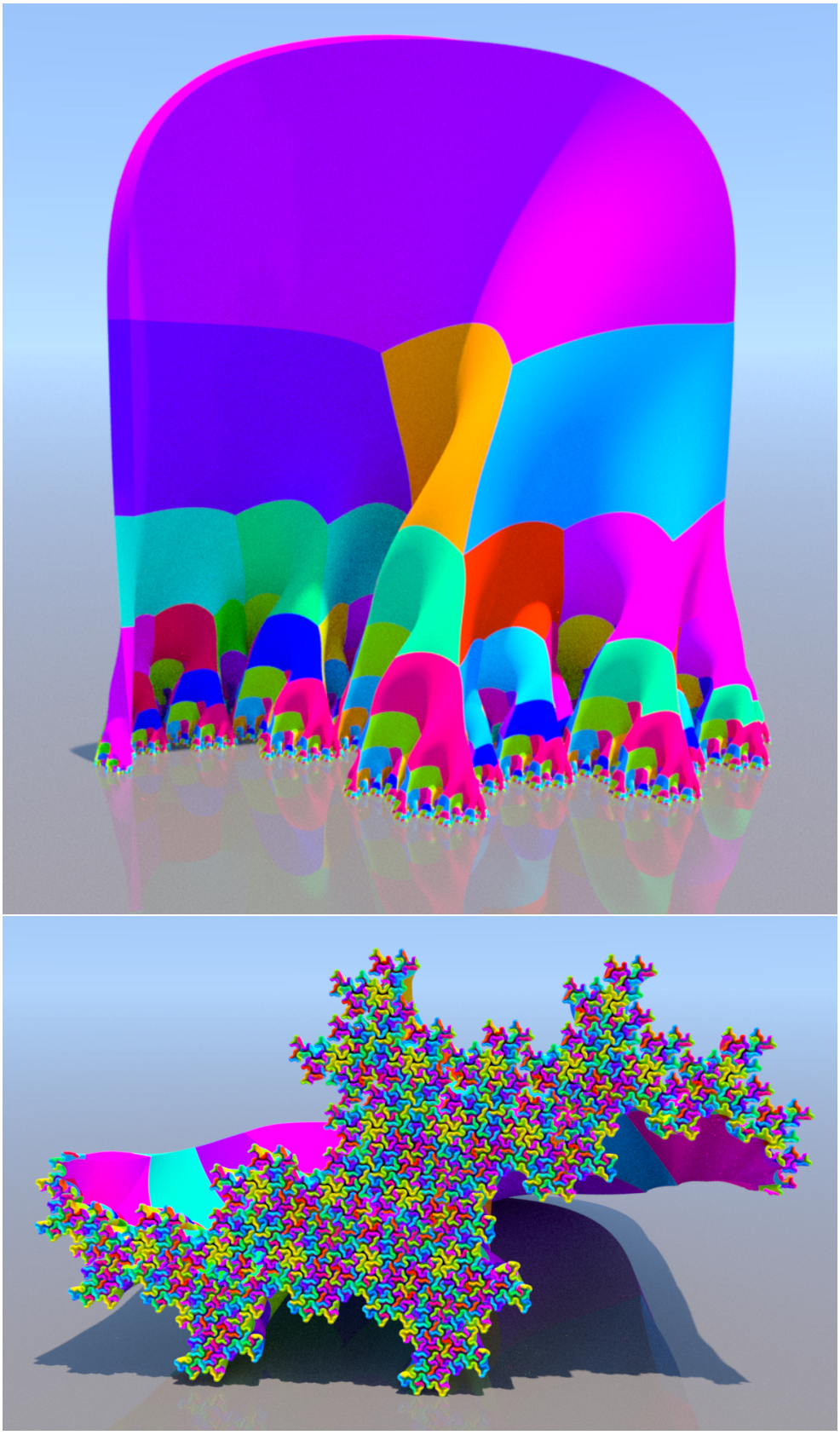


Figure 20: Renders of the developing terdragon curve up to levels 10 (top) and 8 (bottom) (constructed by Loop subdivision).

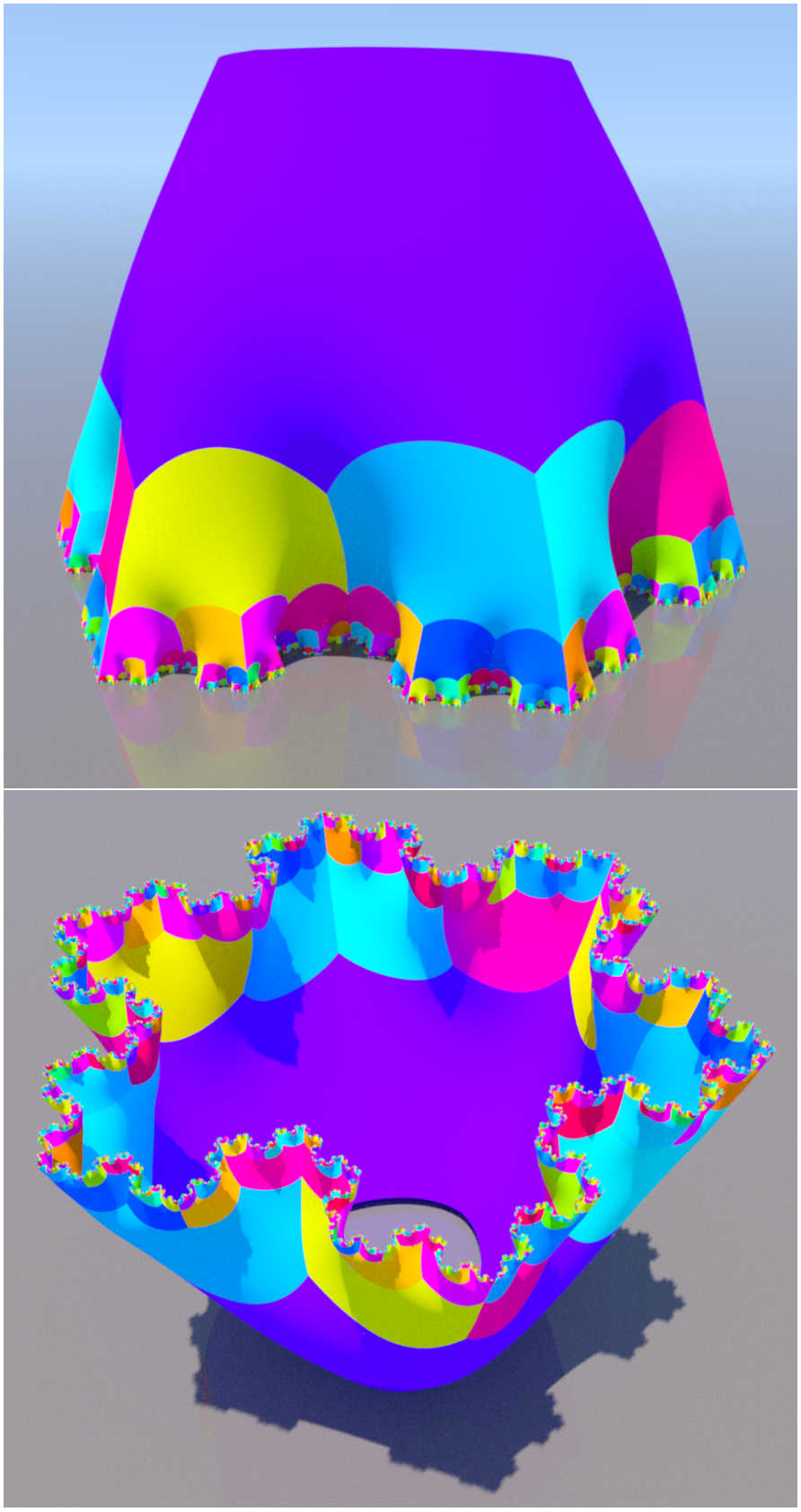


Figure 21: Renders of the developing Koch snowflake up to level 6 (constructed by Loop subdivision).

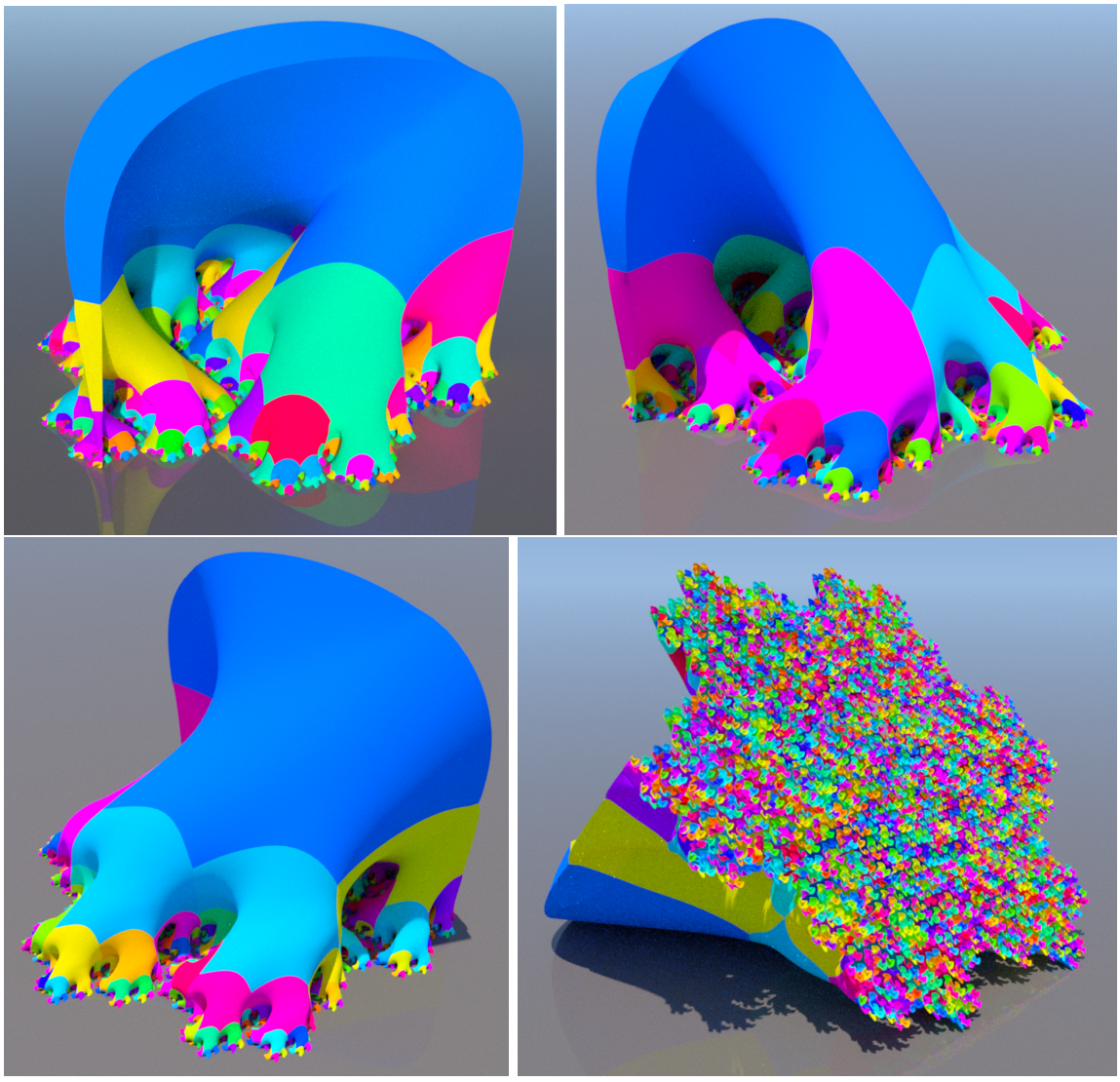


Figure 22: Renders of the developing Gosper curve up to level 5 (constructed by Loop subdivision).

[10] Richard Riesenfeld and Jeffrey Lane, *A theoretical development for the computer generation and display of piecewise polynomial surfaces.*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2(1), 35–46 (1980).

[11] Aristid Lindenmayer, *Mathematical models for cellular interaction in development, Parts I and II*, Journal of Theoretical Biology **18** (1968), 280–315.

[12] Charles Loop, *Smooth Subdivision Surfaces Based on Triangles*, M.S. Mathematics thesis, University of Utah (1987).

[13] _____, *Triangle mesh subdivision with bounded curvature and the convex hull property*, Tech. Rep. MSR-TR-2001-24, Microsoft Research (2001).

[14] B. B. Mandelbrot, *The Fractal Geometry of Nature*, New York: W. H. Freeman, p. 142, 1983.

[15] Przemyslaw Prusinkiewicz, *Graphical applications of L-systems*, Proceedings of Graphics Interface 1986 / Vision Interface 1986, 247–253.

[16] Carlo H. Séquin, *Hilbert Cube 512*, `http://www.cs.berkeley.edu/~sequin/SCULPTS/CHS_bronzes/Hilbert512`, Fabricated by Ex One Company, Prometal division, July 2005.