Into the Portal: Directable Fractal Self-Similarity

Alexa Schor
Yale University
New Haven, CT, USA
alexa.schor@yale.edu

Theodore Kim
Yale University
New Haven, CT, USA
theodore.kim@yale.edu

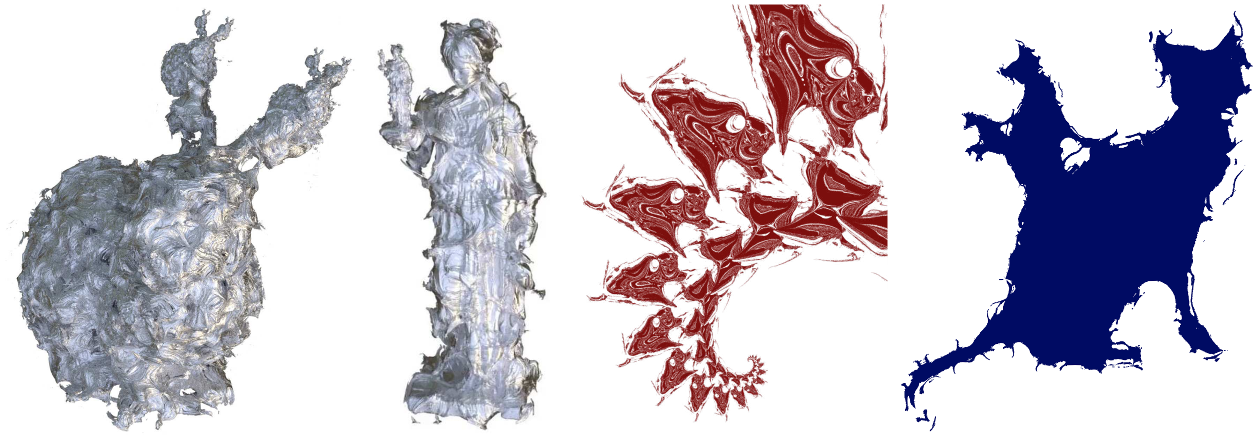
Figure 1: Four self-similar fractals produced by our method. Left to right: fractal bunny with bunnies stacked on the ear tips; fractal *Hebe* recursing atop the bowl in her hand; detailed turbulence in a fractal fox's infinite tails; a stack of fractal cats.

ABSTRACT

We present a novel, directable method for introducing fractal self-similarity into arbitrary shapes. Our method allows a user to directly specify the locations of self-similarities in a Julia set, and is general enough to reproduce other well-known fractals such as the Koch snowflake. Ours is the first algorithm to enable this level of general artistic control while also maintaining the character of the original fractal shape. We introduce the notion of placing "portals" in the iteration space of a dynamical system, bridging the aesthetics of iterated maps with the fine-grained control of iterated function systems (IFS). Our method is effective in both 2D and 3D.

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**; **Shape modeling**; **Parametric curve and surface models**.

KEYWORDS

Julia set shape control, Shape Modulus Julia sets, iterated maps

ACM Reference Format:

Alexa Schor and Theodore Kim. 2024. Into the Portal: Directable Fractal Self-Similarity. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3641519.3657466

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657466

1 INTRODUCTION

Self-similarity, the reproduction of geometric structures across different scales, is one of the most beguiling features of fractal geometry. Its appeal has been leveraged in wide-ranging applications such as film [Giardina 2017; Hutchins et al. 2015], architecture [Ouyang et al. 2021], and installation art [Fakharany 2023].

Some fractal algorithms, such as the dynamical maps that produce the Mandelbrot and Julia sets [Mandelbrot 1982], generate this property spontaneously. Others, such as iterated function systems (IFS) [Barnsley 2014], encode this scale invariance directly in their rules. Previously, controlling self-similarity has always come with artistic tradeoffs. Explicit IFS encodings enable more user control, but can generate rigidly symmetric objects that lack a naturalistic appeal [Barnsley et al. 1988; Demko et al. 1985]. In contrast, the dynamical map approach intrinsically generates a naturalistic look, but their very spontaneity makes the self-similarity difficult to control.

We present an analytic method that combines the control of IFS with the naturalistic look of dynamical maps. We accomplish this by introducing *portals* into the dynamical map that allow a user to explicitly control where self-similarities will occur. Since the portals live within the dynamical map itself, the characteristically turbulent and naturalistic look of the final fractal geometry is still maintained.

While our approach was designed for Julia sets, it is sufficiently general that it can both reproduce classic fractals such as the Koch

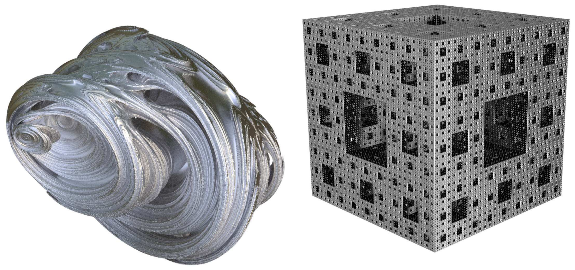
Figure 2: Left: A quaternion Julia set with a value of $c = (-0.2 + 0.4\mathbf{i} - 0.4\mathbf{j} - 0.4\mathbf{k})$. This family of fractals naturally produces appealingly chaotic details, but self-similarities are difficult to control. Right: A Menger [1928] sponge, generated by recursively dividing a cube. The self-similarity is easily controlled, but the shape lacks the chaotic details of the Julia set.

[1904] snowflake, and also open the door to chaotic new variants. We show that portals are effective in both 2D and 3D.

2 BACKGROUND AND RELATED WORK

The dynamical maps that give rise to Julia sets have been known for over a century [Fatou 1917; Julia 1918], and their intricate graphical structures have been studied for over 50 years [Mandelbrot 1980]. One of the most interesting properties of these maps is their self-similarity: the tendency for global structures to reappear at multiple scales [Lei 1990]. While initially formulated for the complex (2D) plane, they were subsequently extended to 3D through the use of quaternions [Hart et al. 1989; Norton 1982].

Julia set fractals such as the one rendered at left of Figure 2 have a unique aesthetic appeal, but controlling them has been an ongoing challenge. Lindsey [2014] showed that by carefully placing the roots of a complex rational, Julia sets could form a fractal approximation of any 2D shape. Subsequent work [Lindsey and Younsi 2019] using Fekete polynomials showed that self-similarity does indeed occur within these shapes. Kim [2015] used a non-linear optimization to perform fractal approximation in 3D and [Schor and Kim 2023] devised an analytic form for the same, but neither work demonstrated that their results were self-similar.

At most, these previous works have generated shapes that *are* self-similar, but never provide a method for *controlling* the size and location of the self-similarity. Users can zoom through the fractal until they happen upon a self-similar duplicate, but tuning the self-similarity to meet external design goals has never been possible.

Other works in graphics have investigated the generation of fractal tilings [Fathauer 2005; Ouyang and Fathauer 2014], including self-similar works inspired by M.C. Escher [Ouyang et al. 2021, 2022]. Generating self-interlocking tiles for "Escherization" [Kaplan and Salesin 2000] has also received considerable attention [Lin et al. 2017; Nagata and Imahori 2023]. These formulations focus on geometric compatibility at a fixed scale, and usually do not attempt to span different scales.

IFS fractals such as the Menger [1928] sponge (Figure 2, right), and grammar-based methods [Wonka et al. 2003] such as L-systems [Prusinkiewicz and Lindenmayer 2012] produce controllable self-similarity, but tend to have the drawback of generating regular, rigid-looking structures. Stochastic versions can be used to introduce variety into plant structures, or in the case of architecture [Merrell 2023], the regularity can be aesthetically desirable. We instead focus on maintaining the turbulent character of dynamical maps, while introducing directable self-similarity into their fractal forms.

3 CREATING SELF-SIMILARITY

In the following, we will denote sets using blackboard $\mathbb{S}$, and their corresponding indicator functions with an argument $\mathbb{S}(\mathbf{x}) \in \{0,1\}$. Points in space are denoted with lowercase bold $\mathbf{x}$, and transformation functions using uppercase bold $\mathbf{F}(\cdot)$.

We will first present our portal technique on the simple case of a Koch [1904] snowflake. Once the basics are established, we will generalize to iterated dynamical maps like those for Julia [1918] sets, as well as the recent reformulation of Schor and Kim [2023].

Once the formulation is complete, we will show that we can now add turbulent details, characteristic of the dynamical map approach, into our original Koch snowflake example.

3.1 Recursive Set Membership

Let $\mathbb{S}(\mathbf{x}) \in \{0,1\}$ be the indicator function for a set $\mathbb{S}$ in $\mathbb{R}^2$. For illustrative purposes, in Figure 3 (left) we show a simple triangle, where black regions denote $\mathbf{x} \in \mathbb{S}$.

In order to add a self-similarity, we will first select a *mask* set $\mathbb{M}$ which denotes where in space the self-similarity should appear. This is shown as the orange shaded region in Figure 3 (middle). We then specify what other regions should be self-similar by devising a transformation $\mathbf{T}(\mathbf{x})$ that maps each point in $\mathbb{M}$ to its corresponding self-similar point. $\mathbf{T}(\mathbf{x})$ is shown with red arrows in Figure 3, and the blue region shows the image of $\mathbb{M}$ under $\mathbf{T}(\mathbf{x})$. In later figures with multiple distinct orange regions, each is mapped to the blue region under $\mathbf{T}(\mathbf{x})$.

To introduce our desired self-similarity into the set function, we build $\mathbb{S}_\mathbb{S}$, a self-similar version of $\mathbb{S}$, where points $\mathbf{x} \in \mathbb{M}$ are first transformed according to $\mathbf{T}(\mathbf{x})$, and then evaluated for membership in $\mathbb{S}$. The transformation $\mathbf{T}(\mathbf{x})$ can be arbitrary as long as it is defined for all points $x \in \mathbb{M}$, but to create self-similarity, it should map onto the geometric structure that the user wants to repeat inside $\mathbb{S}$. In Figure 3 (middle) we chose $\mathbf{T}(\mathbf{x})$ to be a simple scaling and rotation, mapping the orange region to the blue region. We call these constructions *portals* because of the way that they teleport points from an "input region" to a similarly-shaped "output region". The indicator function for Figure 3 (middle) $\mathbb{S}_\mathbb{S}$ is then:

$$\mathbb{S}_\mathbb{S}(\mathbf{x}) = \begin{cases} \mathbb{S}(\mathbf{x}) & \mathbf{x} \notin \mathbb{M} \\ \mathbb{S}(\mathbf{T}(\mathbf{x})) & \mathbf{x} \in \mathbb{M}. \end{cases} \tag{1}$$

This function only makes a single copy of the original triangle and yields a relatively pedestrian two-triangle shape. However, recursive self-similarity can be obtained if we allow this indicator

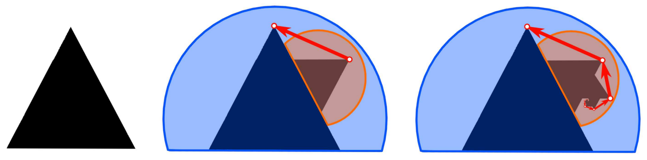

Figure 3: Left: the base set $\mathbb{M}$. Center: evaluating Equation 1 produces a single scaled, rotated copy of the base set. Right: evaluating Equation 2 produces nested self-similar copies. At center and right, arrows show the path of points under transformation $\mathbf{x} \to \mathbf{T}(\mathbf{x})$

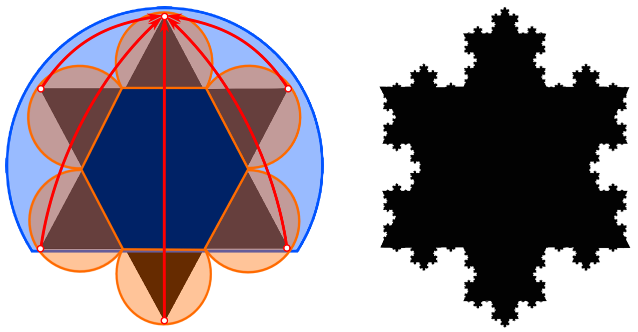

Figure 4: Left: A fractal using a hexagram base set and six portals placed around it. Red arrows show the transformation T. Right: Evaluating $\mathbb{S}_{\text{sim}}$ as described in Equation 2 reproduces the classic Koch [1904] snowflake.

function to recurse on itself:

$$\mathbb{S}_{\mathbb{S}}(\mathbf{x}) = \begin{cases} \mathbb{S}(\mathbf{x}) & \mathbf{x} \notin \mathbb{M} \\ \mathbb{S}_{\mathbb{S}}(\mathbf{T}(\mathbf{x})) & \mathbf{x} \in \mathbb{M} \end{cases}. \tag{2}$$

By allowing $\mathbf{T}(\mathbf{x})$ to recursively transform some points under iteration, self-similarity across scales emerges and one limb of the Koch snowflake is generated in Figure 3 (right). A recursive operation is achieved whenever some points $\mathbf{T}(\mathbf{x})$ for $\mathbf{x} \in \mathbb{M}$ are themselves members of $\mathbb{M}$. There may exist some points $\mathbf{x}$ for which the recursion $\mathbb{S}_{\mathbb{S}}(\mathbf{x})$ does not terminate. While under the above definition set membership is undefined for these points, for consistency with Julia set computation (see §3.2) we consider these to be members of the overall set $\mathbb{S}_{\mathbb{S}}$. However, the self-similarities we describe appear regardless of our treatment of such points.

Our formulation does not restrict $\mathbb{M}$ to a single contiguous region, and $\mathbf{T}(\mathbf{x})$ can be chosen arbitrarily, so the full snowflake can be constructed by placing multiple portals at the same scale (Figure 4, left) and assigning an appropriate $\mathbf{T}(\mathbf{x})$ that maps each orange portal to the same blue shaded region.

We now have a dynamical map that allows us to directly specify self-similar regions. However, it currently only yields the rigid-looking results commonly found in shapes produced by iterated function systems (IFS) [Barnsley 2014], which is reflected in the fact that we have only used $\mathbf{T}(\mathbf{x})$ that correspond to affine transforms or piecewise combinations of the same.

3.2 Adding Portals to Julia Sets

We aim to introduce equivalent self-similarity controls into the chaotic, naturalistic forms generated by dynamical system fractals. To achieve this, we will now examine how to apply our approach to Julia sets. We start with an overview of Julia sets before showing how their computation naturally supports portals.

Filled Julia sets are all the points $\mathbf{x} \in \mathbb{C}$ whose magnitudes remain bounded under the recursive evaluation of some transformation $\mathbf{F}(\mathbf{x}) : \mathbb{C} \to \mathbb{C}$. We can express n recursive applications of $\mathbf{F}(\mathbf{x})$ using a subscript n, such that $\mathbf{F}_n(\mathbf{x})$ denotes

$$\mathbf{F}_1(\mathbf{x}) = \mathbf{F}(\mathbf{x}) \qquad \mathbf{F}_2(\mathbf{x}) = \mathbf{F}(\mathbf{F}(\mathbf{x}))$$
$$\mathbf{F}_3(\mathbf{x}) = \mathbf{F}(\mathbf{F}(\mathbf{F}(\mathbf{x}))) \qquad \mathbf{F}_4(\mathbf{x}) = \mathbf{F}(\mathbf{F}(\mathbf{F}(\mathbf{F}(\mathbf{x})))) \qquad \ldots$$

The filled Julia set $\mathbb{J}$ can then be is defined as

$$\mathbb{J} = \left\{ \mathbf{x} : \lim_{n\to\infty} \|\mathbf{F}_n(\mathbf{x})\| \not\to \infty \right\},$$

where $\|\cdot\|$ denotes magnitude, also known as the modulus. If the point $\mathbf{x}$ does not escape to ∞ after many iterations, then $\mathbf{x} \in \mathbb{J}$. In practice, n is bounded to some large number of iterations (~100), and instead of checking whether the magnitude has grown to infinity, some finite radius is used [Sanders and Kandrot 2010]. Algorithm 1 shows this approach for computing set membership.

ALGORITHM 1: Determine if $\mathbf{x}$ is inside the Julia set of a map $\mathbf{F}$

```
Function IsInsideJuliaSet(x, F)
    i ← 0
    while ||x|| < MaxRadius and i < MaxIterations do
        x ← F(x)
        i ← i + 1
    end
    if i == MaxIterations then return true
    return false
```

Julia sets were originally computed on the complex numbers, where $\mathbf{x} \in \mathbb{C}$ and $\mathbf{F} : \mathbb{C} \to \mathbb{C}$, but this same computation can be performed in any space. The most common choice of $\mathbf{F}$ is the quadratic map $\mathbf{F}(\mathbf{x}) = \mathbf{x}^2 + \mathbf{c}$, where $\mathbf{c}$ is an arbitrary constant that determines the global shape of the Julia set [Mandelbrot 1982]. For 2D Julia sets this is conducted on $\mathbb{C}$, while for most 3D Julia sets iteration is conducted on the quaternions $\mathbb{H}$ with a 3D slice then extracted, as in Figure 2, left. Despite the popularity of the quadratic form, any map $\mathbf{F}(\mathbf{x})$ can be used, so we will design our own now in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Unlike with our Koch snowflake $\mathbb{S}$, the definition of the Julia set $\mathbb{J}$ is *already* recursive. Therefore, instead of using $\mathbb{M}$ and $\mathbf{T}(\mathbf{x})$ to define a new recursive indicator function $\mathbb{S}_{\mathbb{S}}$ (Eqn. 2), we can instead introduce portals directly into the $\mathbf{F}(\mathbf{x})$ map:

$$\mathbf{F}_{\mathbf{F}}(\mathbf{x}) = \begin{cases} \mathbf{F}(\mathbf{x}) & \mathbf{x} \notin \mathbb{M} \\ \mathbf{T}(\mathbf{x}) & \mathbf{x} \in \mathbb{M} \end{cases} \tag{3}$$

Under Julia set iteration, the set membership of any point $\mathbf{x}$ can be determined by any point $\mathbf{F}_n(\mathbf{x})$ along its iteration *without* any knowledge of its prior iterates. Because of this, mapping the iteration of one point $(\mathbf{x})$ to another $(\mathbf{y})$ will ensure an equivalence of set membership: $(\mathbf{F}_n(\mathbf{x}) = \mathbf{y}) \implies (\mathbb{J}(\mathbf{x}) = \mathbb{J}(\mathbf{y}))$, as illustrated in Figure 5 Adding portals to a dynamical map creates the left-hand-side of the conditional. User-specified components are then repeated, just as portions of the indicator function were duplicated in Section 3.1. This occurs infinitely, without any additional modifications, as Julia set computations are intrinsically recursive.

Thus, introducing portals into the dynamical map keeps their characteristic chaotic details intact, but also adds a level of control over self-similarity that is more reminiscent of an IFS. Figure 6 shows our technique applied to the "rabbit" Julia set of Douady et al. [1984]. The overall shape and chaotic details remain intact, but new self-similarities have been added where desired. This operation also reveals regions of self-similarity embedded in the original dynamical map, as new copies of the "rabbit" spontaneously appear in regions that were not directly specified by our portals, but that map into our portals under some $\mathbf{F}_n(\mathbf{x})$.

3.3 Extension to 3D Shape Modulus Formulation

The preceding examination of Julia sets was for the 2D complex plane ($\mathbf{F}(\mathbf{x}) : \mathbb{C} \rightarrow \mathbb{C}$), and the most common extension to 3D is to perform similar computations over 4D quaternions ($\mathbf{F}(\mathbf{x}) : \mathbb{H} \rightarrow \mathbb{H}$ [Norton 1982]) and extract a 3D slice. To add self-similarity, the portal computation in Equation 3 can be carried over directly to Julia sets in any dimension.

However, both the complex and quaternion dynamical maps have a known limitation: they are notoriously difficult to control. While our portals can be used to introduce self-similarity into any portion of them, persuading them to generate the desired "base case" shape to repeat, such as a fractal version of the original triangle in Figure 3, is very difficult. Analytic methods exist in 2D [Lindsey and Younsi 2019], but only numerical methods are known in 3D [Kim 2015].

Fortunately, a recently proposed *shape modulus* approach [Schor and Kim 2023] both introduces better user controls into these dynamical maps, and fits naturally into our portal formulation. In that approach, an original quaternion dynamical map is first decomposed into modulus ($\mathbf{R}(\mathbf{x}) \in \mathbb{R}$) and versor ($\mathbf{D}(\mathbf{x}) \in \mathbb{R}^3$) components, otherwise known as magnitude and direction

$$\mathbf{R}(\mathbf{x}) = ||\mathbf{F}(\mathbf{x})|| \qquad \mathbf{D}(\mathbf{x}) = \frac{\mathbf{F}(\mathbf{x})}{||\mathbf{F}(\mathbf{x})||}. \tag{4}$$

The $\mathbf{R}(\mathbf{x})$ is then replaced with an alternative $\hat{\mathbf{R}}(\mathbf{x})$ that explicitly incorporates the signed distance field $\phi(\mathbf{x})$ of a desired shape:

$$\hat{\mathbf{R}}(\mathbf{x}) = e^{\alpha(\phi(\mathbf{x})+\beta)} \qquad \mathbf{D}(\mathbf{x}) = \frac{\mathbf{F}(\mathbf{x})}{||\mathbf{F}(\mathbf{x})||}. \tag{5}$$

The α and β constants are "thinness" and "offset" control variables [Schor and Kim 2023]. The dynamical map then becomes $\hat{\mathbf{F}}(\mathbf{x}) = \hat{\mathbf{R}}(\mathbf{x})\mathbf{D}(\mathbf{x})$ in lieu of the original $\mathbf{F}(\mathbf{x})$.

The portal transform can then be applied:

$$\hat{\mathbf{F}}_{\mathrm{F}}(\mathbf{x}) = \begin{cases} \hat{\mathbf{F}}(\mathbf{x}) & \mathbf{x} \notin \mathbb{M} \\ \mathbf{T}(\mathbf{x}) & \mathbf{x} \in \mathbb{M} \end{cases}. \tag{6}$$

This formulation enables considerable flexibility: all the control parameters from the shape modulus approach are inherited by our portals. Additionally, because the shape modulus formulation decouples 3D Julia set computation from quaternion polynomial evaluation, we are able to conduct iteration in $\mathbb{R}^3$.

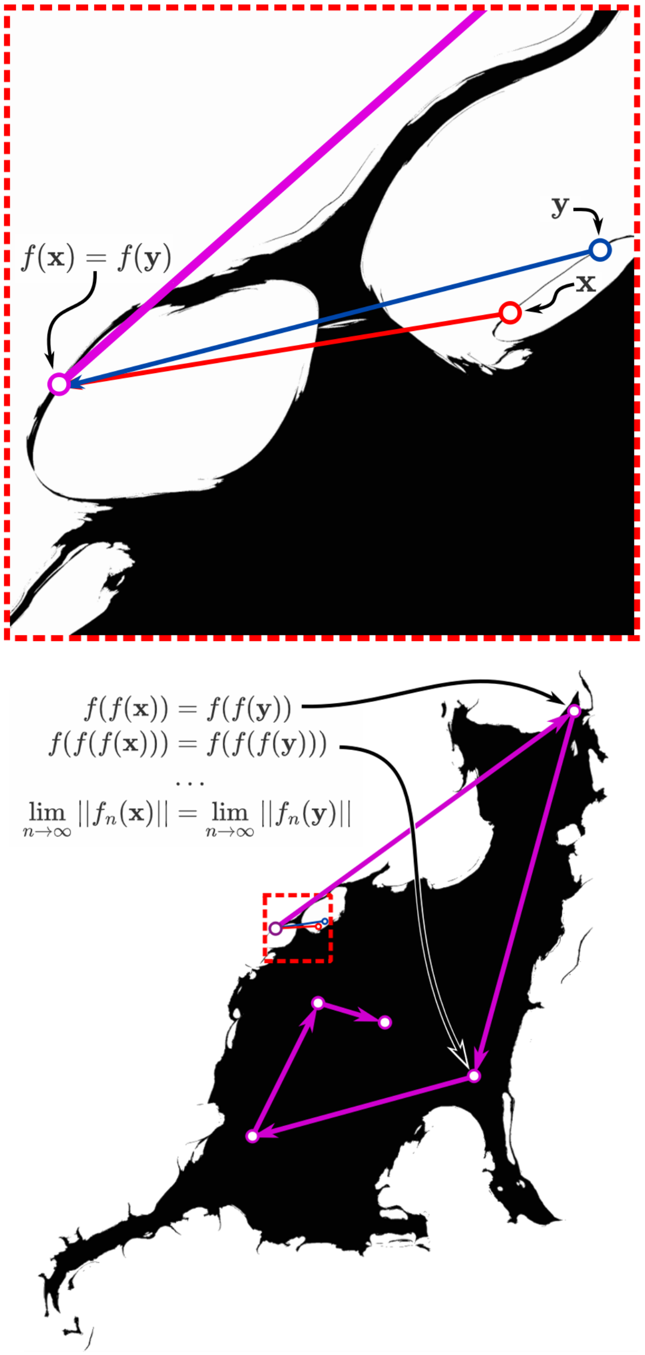
$f(\mathbf{x}) = f(\mathbf{y})$
$\mathbf{y}$
$\mathbf{x}$
$f(f(\mathbf{x})) = f(f(\mathbf{y}))$
$f(f(f(\mathbf{x}))) = f(f(f(\mathbf{y})))$
$\cdots$
$\lim_{n\to\infty} ||f_n(\mathbf{x})|| = \lim_{n\to\infty} ||f_n(\mathbf{y})||$

Figure 5: Top: A zoomed view of a 2D cat-shaped Julia set $\mathbb{J}(f)$, with transformation $f(\cdot)$ shown with colored arrows. Two different points, x and y, map to the same point, $f(\mathbf{x}) = f(\mathbf{y})$. Bottom: under Julia set computation, all subsequent iterations and eventual set membership of the two iterates become thereafter identical. This property of Julia sets is what allows the portal mechanism described in §3.2 to produce self-similarities.

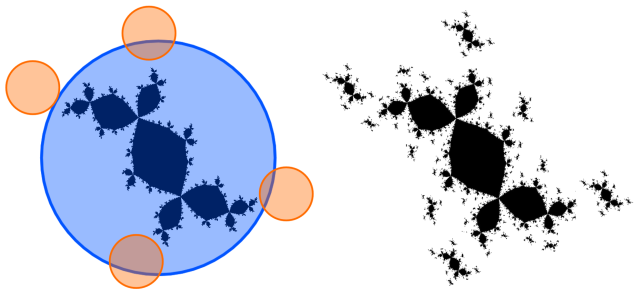

Figure 6: Left: the "rabbit" Julia set of Douady et al. [1984], with mask shown in blue and portals in orange, similar to Fig. 4. Right: The rabbit with our new self-similarities added.

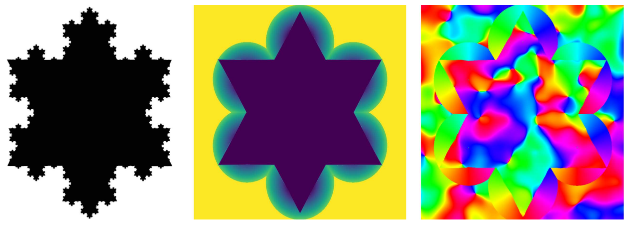

Figure 7: Left: The Koch [1904] snowflake, reproduced using Eqn. 6, the portal placements from Fig. 4, and a large α. Center and right: the modulus (magnitude) and versor (direction) fields.

One of the hallmarks of dynamical maps is *quasi self-similarity* [Järvi 1997], where repeating structures are never quite exact copies of the original. We can precisely control the level of quasi self-similarity by distorting $\mathbf{T}$, or, alternatively, by introducing a scalar blend factor $\mathbf{B}(\mathbf{x}) \in \mathbb{R}$ into the application of $\mathbf{T}(\mathbf{x})$:

$$\hat{\mathbf{F}}_{\mathrm{F}}(\mathbf{x}) = \begin{cases} \hat{\mathbf{F}}(\mathbf{x}) & \mathbf{x} \notin \mathbb{M} \\ \mathbf{B}(\mathbf{x}) \cdot \mathbf{T}(\mathbf{x}) + (1 - \mathbf{B}(\mathbf{x})) \cdot \hat{\mathbf{F}}(\mathbf{x}) & \mathbf{x} \in \mathbb{M} \end{cases}. \tag{7}$$

For $\mathbf{B}(\mathbf{x}) = 1$, we recover Eqn. 6, and $\mathbf{B}(\mathbf{x}) = 0$ produces the original base Julia set without portals. By spatially varying $\mathbf{B}(\mathbf{x})$, a smooth transition between portal and non-portal regions can be achieved. For the 2D and 3D examples, a $\mathbf{B}(\mathbf{x})$ was chosen that slightly feathered the edges of the mask set.

Putting everything together, we can now generate a "turbulent" version of the Koch snowflake [Fincher 1999]. By setting α to be large, we first reproduce the original snowflake (Fig. 7). If we then lower α and set the surrounding versor field according to the gradient of a noise function (e.g. Perlin [1985]), we can gradually introduce the chaotic details characteristic of Julia sets (Fig. 8). For this example, we set $\mathbb{M}$ as the set difference of circular portal regions and the base Julia set, ensuring that the self-similar fractal detail functions as an additive operation onto the base Julia set.

4 RESULTS

We have applied our algorithm to a variety of examples in both 2D and 3D. As much of the visual appeal of Julia sets appears in 2D, we have applied portals to a pair of 2D examples.

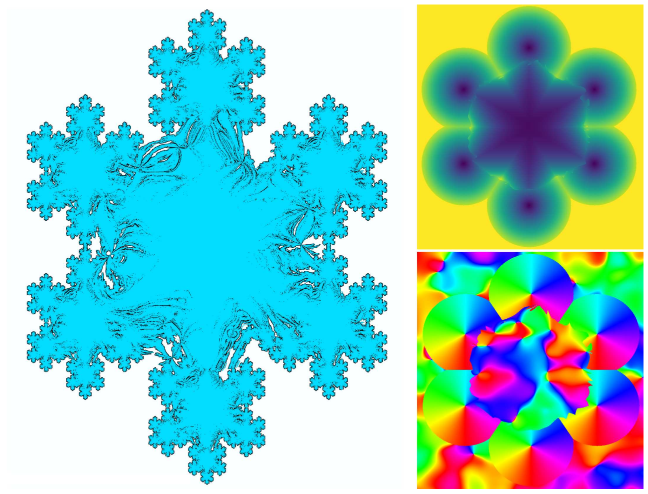

Figure 8: Left: A "turbulent" Koch [1904] snowflake generated by lowering the α in Eqn. 5. Right: The modulus (top) and versor (bottom) fields.

In Figure 9, in a nod to the mythical nine-tailed fox, we have added a portal that generates an ∞-*tailed fox*. The algorithm is quite efficient, and the shape only takes 30 seconds to compute on a single-core machine (Table 1). Similar to the Koch snowflake, by adjusting the α parameter, gradually more turbulent versions of the fox can be generated (Figure 9, right). Zooming in reveals intricate details that we did not originally specify, such a spontaneously generated quasi self-similar marbling pattern (Figure 11, left).

In Figure 10, in a nod to "turtles all the way down" [Hawking 2009], we have introduced a portal into the cat model from Lindsey [2014] and generated "cats all the way down". Like the fox model, the computation is quite efficient and completes in slightly more than 30 seconds (Table 1). We again are able to introduce turbulence into the shape by adjusting α. The intricate details that appear when zooming in to the portal are shown on the right of Figure 11. Similar to the spontaneous self-similarites that appear in Lindsey and Younsi [2019], we found that zooming into the cat revealed spontaneous copies of the cat's head, even in regions that we did not indicate with our portals. This phenomenon is similar to Figure 6, where the portals induced new copies of the Douady et al. [1984] "rabbit" to appear throughout space.

Turning to 3D, we generate a self-similar version of a *Hebe* garden statue by placing a portal on the bowl that the statue holds (Figure 13). We again generate a turbulent version of the model by adjusting both the α and β parameters. The resultant geometry is extremely intricate, so we use non-linear marching cubes [Kim 2015] to extract it. While the final model is quite large and complex (~30 million triangles), the natural parallelism of marching cubes allowed us to compute the model in just over 3 minutes. For all 3D examples, the marching cubes domain was divided into a perfect octree of 512 tiles and then processed in parallel on Intel Xeon cores with clock speeds between 2 and 2.9 GHz.

In Figure 12, we introduce multiple portals onto the bunny's ears in order to generate "bunnies all the way down". This is the most intricate 3D model we attempted, which yielded over 300 million

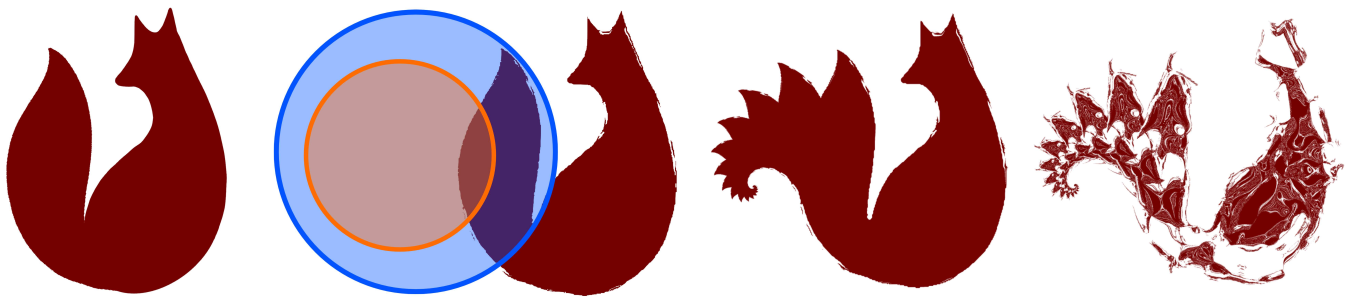

Figure 9: ***An ∞-tailed fox.*** **From left to right: The original (target) fox shape; The shape modulus base fractal with a high α value and location of portals illustrated; The fractal with evaluated those portals present; The fractal evaluated with a lower α value, increasing turbulent detail.**

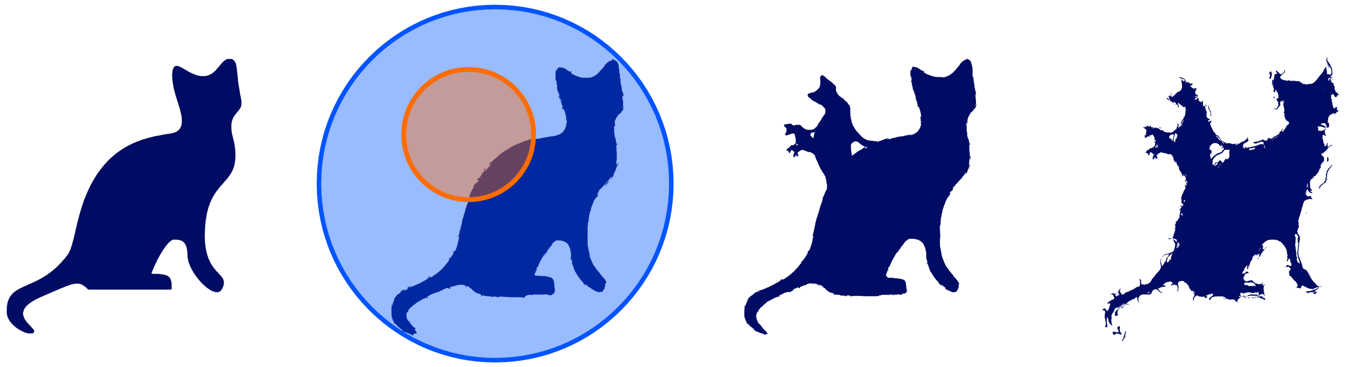

Figure 10: ***Cats all the way down.*** **From left to right: The original cat shape; The shape modulus base fractal with a high α value and location of portals illustrated; The fractal evaluated with those portals present; The fractal evaluated with a lower α value, increasing turbulent detail.**

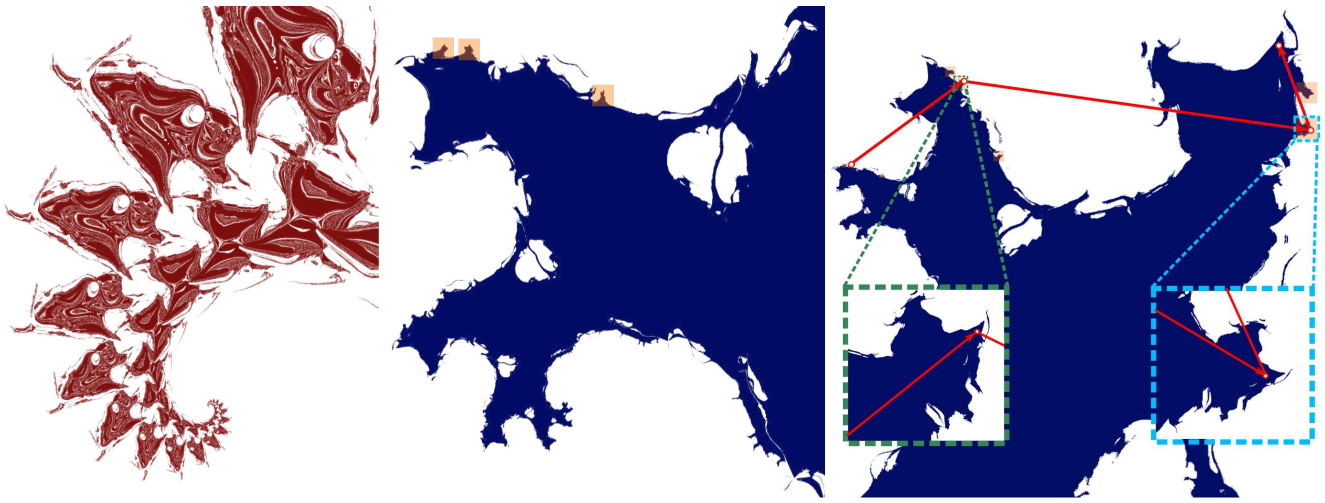

Figure 11: Left: detailed infinite tails from Figure 9. Center: detailed infinite cats from Figure 10. Orange boxes show spontaneous copies of the cat's head. Right: Spontaneous copies arise when iterates coherently map to larger-scale features. The smaller heads map back to the large, original head under iteration.

Table 1: 2D and 3D Performance. For all 3D examples, triangles were computed using marching cubes on a grid resolution of 2400^3. For all 2D examples, a uniform sampling of 1600^2 was used. 3D jobs were divided spatially, so each computed varying amounts of geometry. Thus, wall-clock time differs slightly from total time divided by the number of cores. All times are in hours:minutes:seconds.

Fig.	Example	Dimension	α	β	Portals	Triangles	CPU time	Cores	Wall-clock time
7	Snowflake (low α)	2D	36	-0.7	3	-	00:00:42	1	00:00:42
8	Snowflake (high α)	2D	300	-0.7	3	-	00:00:37	1	00:00:37
9	Fox (low α)	2D	35	-0.70	1	-	00:00:30	1	00:00:30
9	Fox (high α)	2D	300	0.85	1	-	00:00:30	1	00:00:30
10	Cat (low α)	2D	38	-0.75	1	-	00:00:36	1	00:00:36
10	Cat (high α)	2D	300	-0.75	1	-	00:00:37	1	00:00:37
13	Hebe (high α)	3D	0.4	5.8	1	13.5M	6:12:03	512	00:02:39
13	Hebe (low α)	3D	0.27	8.2	1	17.75M	5:49:21	512	00:03:09
12	Bunny	3D	10	0.10	2	339M	66:57:09	512	00:14:57

triangles. As the computation remains trivially parallelizeable, the geometry was extracted in less than 15 minutes.

5 FUTURE WORK

We have shown how to introduce arbitrary self-similarities into dynamical map fractals by introducing the concept of *portals*. Currently, we have only explored a simple set of transformations for these portals, such as affine transforms or affine blended with noise gradients. While these already generate self-similarity, further investigation is needed to determine what new shapes can be generated by more sophisticated transforms.

Our portal-based strategy for inducing self-similarities is effective on Julia sets and for any other system where the property from Figure 3 holds. However, this also means our method cannot be directly applied to systems such as the Mandelbrot set. Different techniques may be required to enable artistic self-similarities in different scenarios.

Currently, portals are placed by hand. While this provides a considerable amount of control to the user, it is not difficult to imagine a different workflow where the self-similarities are automatically placed by detecting the regions of the base shape that would form visually pleasing recursions. However, such a detection algorithm remains future work.

Finally, we have found that the 3D shapes generated by our algorithm can contain more details than the underlying non-linear marching cubes algorithm is able to resolve. This is true of most fractal shapes by their very fractal nature, but especially relevant here when introducing self-similarities across scales. We have found that a dual contouring approach [Ju et al. 2002] does not produce improved results, as the surface normals change very rapidly in the regions with the richest fractal detail. A robust, fractal-aware meshing algorithm remains future work.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable feedback and for taking a chance on an offbeat topic. This work was supported by Adobe and the Teng and Han Family Fund.

REFERENCES

Michael Barnsley. 2014. *Fractals everywhere* (2nd ed.). Academic Press.

Michael F Barnsley, Arnaud Jacquin, Francois Malassenet, Laurie Reuter, and Alan D Sloan. 1988. Harnessing chaos for image synthesis. In *Proceedings of SIGGRAPH*. 131–140.

Stephen Demko, Laurie Hodges, and Bruce Naylor. 1985. Construction of Fractal Objects with Iterated Function Systems. In *Proceedings of SIGGRAPH*. 271–278.

Adrien Douady, John Hamal Hubbard, and Pierre Lavaurs. 1984. *Etude dynamique des polynômes complexes*. Université de Paris-Sud, Dép. de Mathématique Orsay, France.

Nour Fakharany. 2023. Jim Denevan's Monumental Land Art Debutes in Abu Dhabi. *ArchDaily* (Nov. 2023).

Robert W Fathauer. 2005. Fractal tilings based on dissections of polyhexes. In *Proceedings of Bridges*. 427–434.

Pierre Fatou. 1917. Sur les substitutions rationnelles. *Comptes Rendus de l'Académie des Sciences de Paris* 164 (1917), 806–808.

David Fincher. 1999. Fight Club.

Carolyn Giardina. 2017. 'Guardians of the Galaxy Vol. 2': A Digital Kurt Russell and Other VFX Tricks Revealed. *The Hollywood Reporter* (May 2017).

John C Hart, Daniel J Sandin, and Louis H Kauffman. 1989. Ray tracing deterministic 3-D fractals. In *Proceedings of SIGGRAPH*. 289–296.

Stephen Hawking. 2009. *A brief history of time: from big bang to black holes*. Random House.

David Hutchins, Olun Riley, Jesse Erickson, Alexey Stomakhin, Ralf Habel, and Michael Kaschalk. 2015. Big Hero 6: into the portal. In *SIGGRAPH Talks*. 1–1.

Pentti Järvi. 1997. Not all Julia sets are quasi-self-similar. *Proc. Amer. Math. Soc.* 125, 3 (1997), 835–838.

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of hermite data. In *Proceedings of SIGGRAPH*. 339–346.

Gaston Julia. 1918. Mémoire sur l'itération des fonctions rationnelles. *J. Math. Pures Appl.* 8 (1918), 47–245.

Craig S. Kaplan and David H. Salesin. 2000. Escherization. In *Proceedings of SIGGRAPH*. 499–510.

Theodore Kim. 2015. Quaternion Julia Set Shape Optimization. In *Proceedings of the Eurographics Symposium on Geometry Processing*. 167–176.

HV Koch. 1904. Sur une courbe continue sans tangente, obtenue par une construction géométrique élémentaire. *Arkiv for Matematik, Astronomi och Fysik* 1 (1904), 681–704.

Tan Lei. 1990. Similarity between the Mandelbrot set and Julia sets. *Communications in mathematical physics* 134 (1990), 587–617.

Shih-Syun Lin, Charles C Morace, Chao-Hung Lin, Li-Fong Hsu, and Tong-Yee Lee. 2017. Generation of escher arts with dual perception. *IEEE transactions on visualization and computer graphics* 24, 2 (2017), 1103–1113.

Kathryn Lindsey and Malik Younsi. 2019. Fekete polynomials and shapes of Julia sets. *Trans. Amer. Math. Soc.* 371, 12 (2019), 8489–8511.

Kathryn A. Lindsey. 2014. Shapes of polynomial Julia sets. *Ergodic Theory and Dynamical Systems* 35, 6 (Aug 2014), 1913–1924. https://doi.org/10.1017/etds.2014.8

Benoit B Mandelbrot. 1980. Fractal aspects of the iteration of $z \longrightarrow \Lambda z$ (1-z) for complex Λ and z. *Annals of the New York Academy of Sciences* 357, 1 (1980), 249–259.

Benoit B Mandelbrot. 1982. *The fractal geometry of nature*. Vol. 1. WH Freeman.

Karl Menger. 1928. *Dimensionstheorie: Karl Menger*. Springer.

Paul Merrell. 2023. Example-Based Procedural Modeling Using Graph Grammars. *ACM Trans. Graph.* 42, 4, Article 60 (jul 2023), 16 pages.

Yuichi Nagata and Shinji Imahori. 2023. Creation of Dihedral Escher-like Tilings Based on As-Rigid-As-Possible Deformation. *ACM Trans. Graph.* (dec 2023).

Alan Norton. 1982. Generation and display of geometric fractals in 3-D. *ACM SIGGRAPH Computer Graphics* 16, 3 (July 1982), 61–67.

Peichang Ouyang, Kwok Wai Chung, Alain Nicolas, and Krzysztof Gdawiec. 2021. Self-Similar Fractal Drawings Inspired by M. C. Escher's Print Square Limit. *ACM Trans. Graph.* 40, 3, Article 31 (jul 2021), 34 pages.

Peichang Ouyang and Robert W Fathauer. 2014. Beautiful math, part 2: aesthetic patterns based on fractal tilings. *IEEE Computer Graphics and applications* 34, 1 (2014), 68–76.

Peichang Ouyang, Krzysztof Gdawiec, Alain Nicolas, David Bailey, and Kwok Wai Chung. 2022. Interlocking Spiral Drawings Inspired by M. C. Escher's Print Whirlpools. *ACM Trans. Graph.* 42, 2, Article 18 (nov 2022), 17 pages.

Ken Perlin. 1985. An image synthesizer. *Proceedings of SIGGRAPH* 19, 3 (1985), 287–296.

Przemyslaw Prusinkiewicz and Aristid Lindenmayer. 2012. *The algorithmic beauty of plants.* Springer Science & Business Media.

Jason Sanders and Edward Kandrot. 2010. *CUDA by example: an introduction to general-purpose GPU programming.* Addison-Wesley Professional.

Alexa Schor and Theodore Kim. 2023. A Shape Modulus for Fractal Geometry Generation. In *Proceedings of the Eurographics Symposium on Geometry Processing.* 9 pages.

Peter Wonka, Michael Wimmer, François Sillion, and William Ribarsky. 2003. Instant Architecture. *ACM Trans. Graph.* 22, 3 (jul 2003), 669–677.

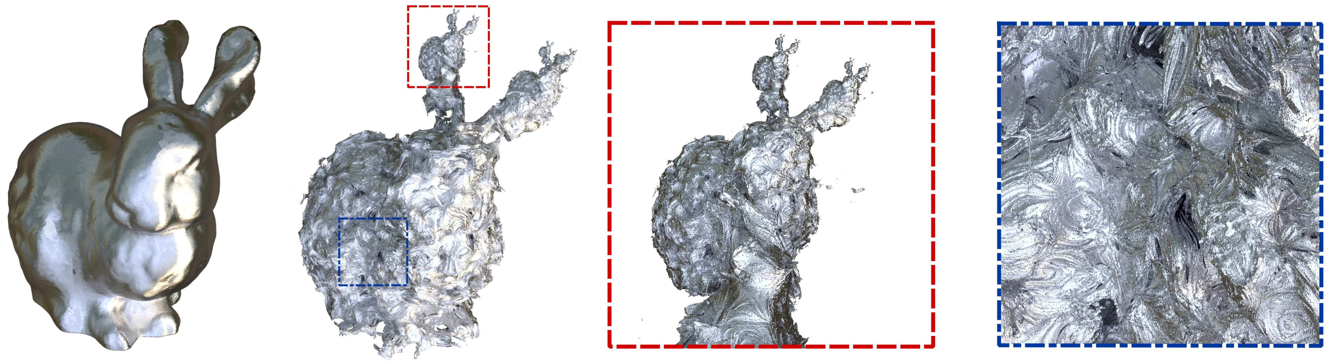

Figure 12: *Bunnies All the Way Down.* Left to right: The original bunny mesh; a self-similar fractal using our technique, where portals are inserted at each ear tip; a zoomed-in view of the self-similar region in the portal on one ear; a zoomed view of the turbulent details on along the bunny's body.

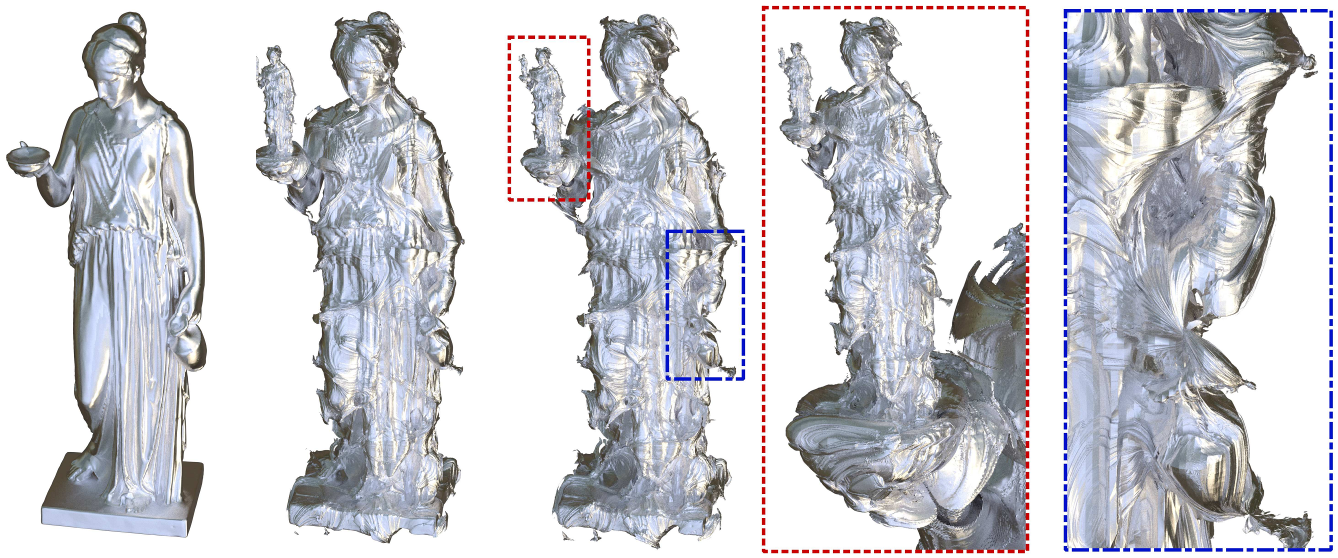

Figure 13: *Fractal Hebe.* Left to right: the original Hebe mesh; a self-similar fractal using our technique with a high value of α yields a light amount of fractal detail; reducing α increases the turbulent fractal detail; a zoomed view of the self-similar region inside the portal; a zoomed-in view of the intricate detail that appears near the statue's base.