The Fractal Nature of Bezier Curves

Ron Goldman
Department of Computer Science
Rice University
6100 Main Street
Houston, Texas 77005-1892
rng@cs.rice.edu

Abstract

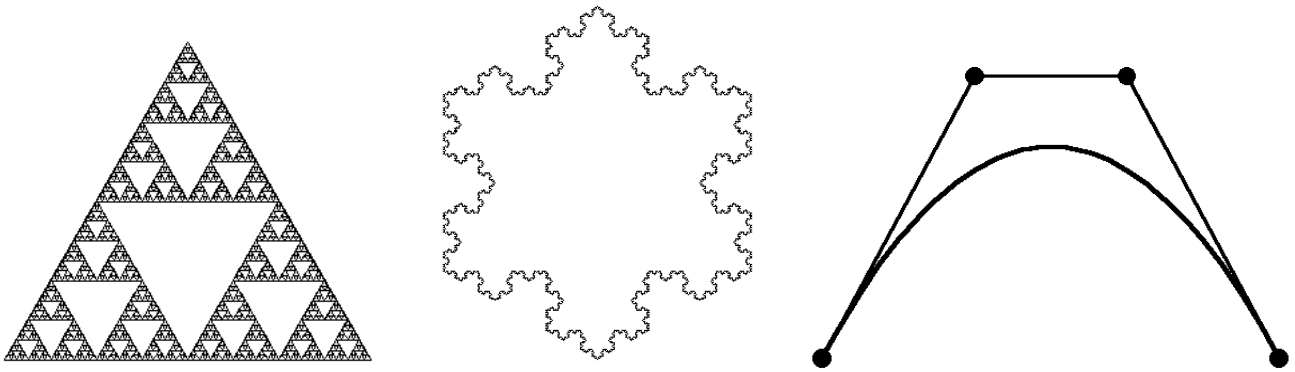
Fractals are attractors -- fixed points of iterated function systems. Bezier curves are polynomials -- linear combinations of Bernstein basis functions. The de Casteljau subdivision algorithm is used here to show that Bezier curves are also attractors. Thus, somewhat surprisingly, Bezier curves are fractals. This fractal nature of Bezier curves is exploited to derive a new rendering algorithm for Bezier curves.

1. Introduction

Fractals are famous both for their strange appearance and for their odd geometric properties. The Sierpinski gasket and the Koch snowflake in Figure 1 are two well known examples of fractal curves. The Koch snowflake is continuous everywhere, but differentiable nowhere; the Sierpinski gasket has Hausdorff dimension log(3)/log(2) -- a fractional dimension greater than one but less than two.

Bezier curves, in contrast, are polynomial curves. Polynomial curves are one-dimensional and infinitely differentiable almost everywhere. Typically then, we do not think of Bezier curves as fractals, since polynomial curves do not appear to possess any of the strange features of fractal curves. Nevertheless, the goal of this paper is to show that Bezier curves are also fractals. We shall exploit this fractal nature of Bezier curves to present a new algorithm for rendering Bezier curves.

To understand how it is that Bezier curves are fractal curves, we need a precise definition of the term *fractal*. We begin then in Section 2 with one commonly accepted definition for fractals as attractors. The signature of an attractor is that convergence to an attractor is independent of the starting set. In Section 3, we apply the de Casteljau subdivision algorithm to show that Bezier curves are attractors, and we derive a novel rendering algorithm for Bezier curves based on the signature property of attractors. We close in Section 4 with a few observations on the fractal nature of Bezier surfaces and some open questions for future research. An Appendix on the Trivial Fixed Point Theorem and its consequences is provided to explain in more detail why convergence to an attractor is independent of the starting set.



(a) Sierpinski Triangle *(b) Koch Snowflake* *(c) Bezier Curve*

Figure 1: Fractals: (a) a Sierpinski triangle, (b) a Koch snowflake, and (c) a Bezier curve.

2. Fractals as Fixed Points of Iterated Function Systems

Fractals are attractors -- fixed points of iterated function systems [1]. The remainder of this section is devoted to explaining this definition and to deriving some of its consequences.

An *iterated function system* consists of a collection of contractive transformations $W = \{w_1, \ldots, w_l\}$, where a map w is said to be *contractive* if for every pair of points P, Q

$$dist\big(w(P), w(Q)\big) \le s\, dist\,(P,Q) \qquad 0 < s < 1.$$

We can apply a map w to a set S by letting

$$w(S) = \{w(x) \mid x \in S\}.$$

Similarly, can apply an iterated function system W to a set S by letting

$$W(S) = w_1(S) \cup \cdots \cup w_l(S).$$

A set S is said to be a *fixed point* of an iterated function system W if $W(S) = S$.

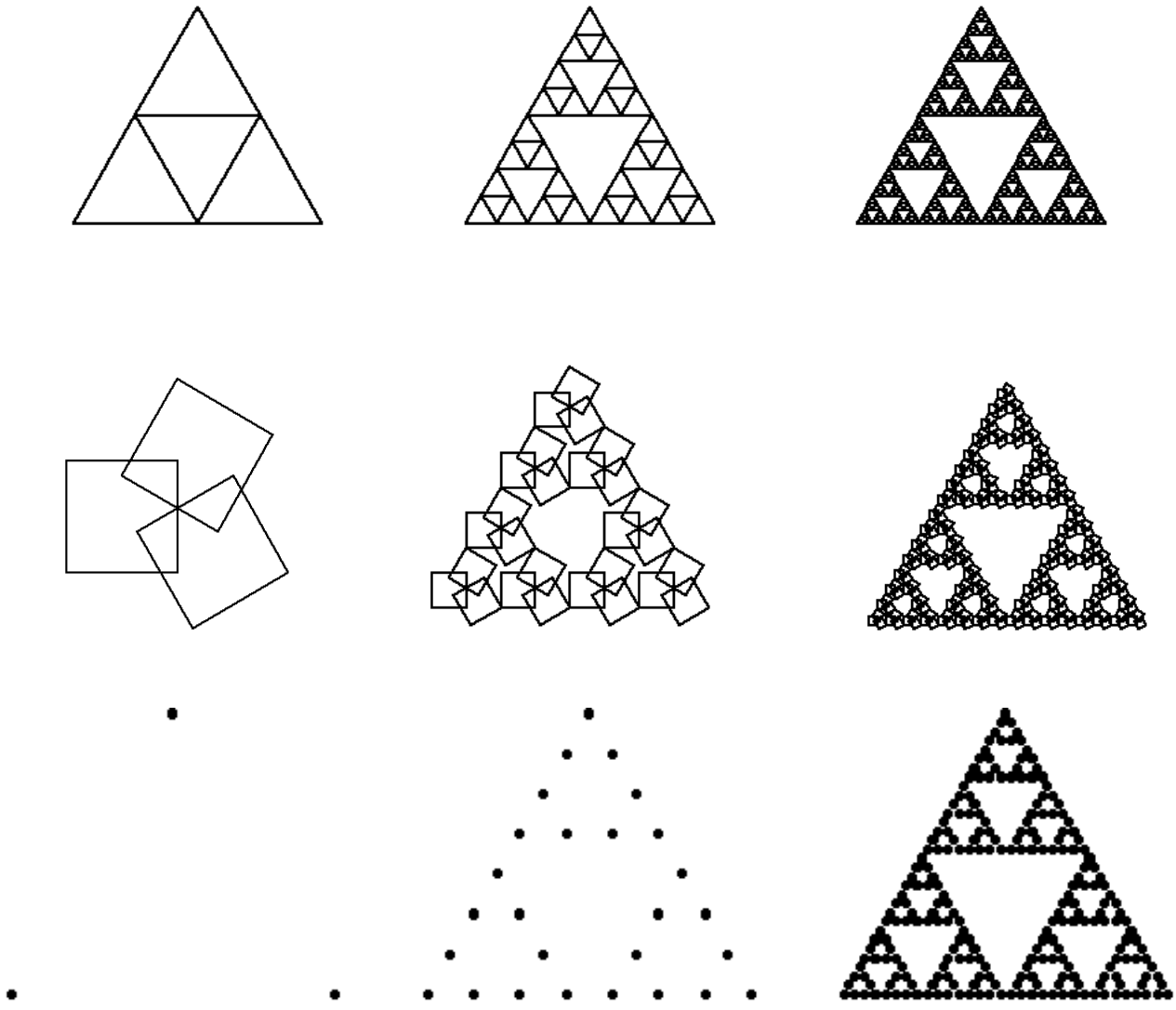


Figure 2: The Sierpinski gasket. The top row is generated starting from an equilateral triangle; the middle row is generated starting from a square instead of a triangle; the bottom row is generated starting from a single point. In each case, levels 1, 3 and 5 of the iteration are illustrated.

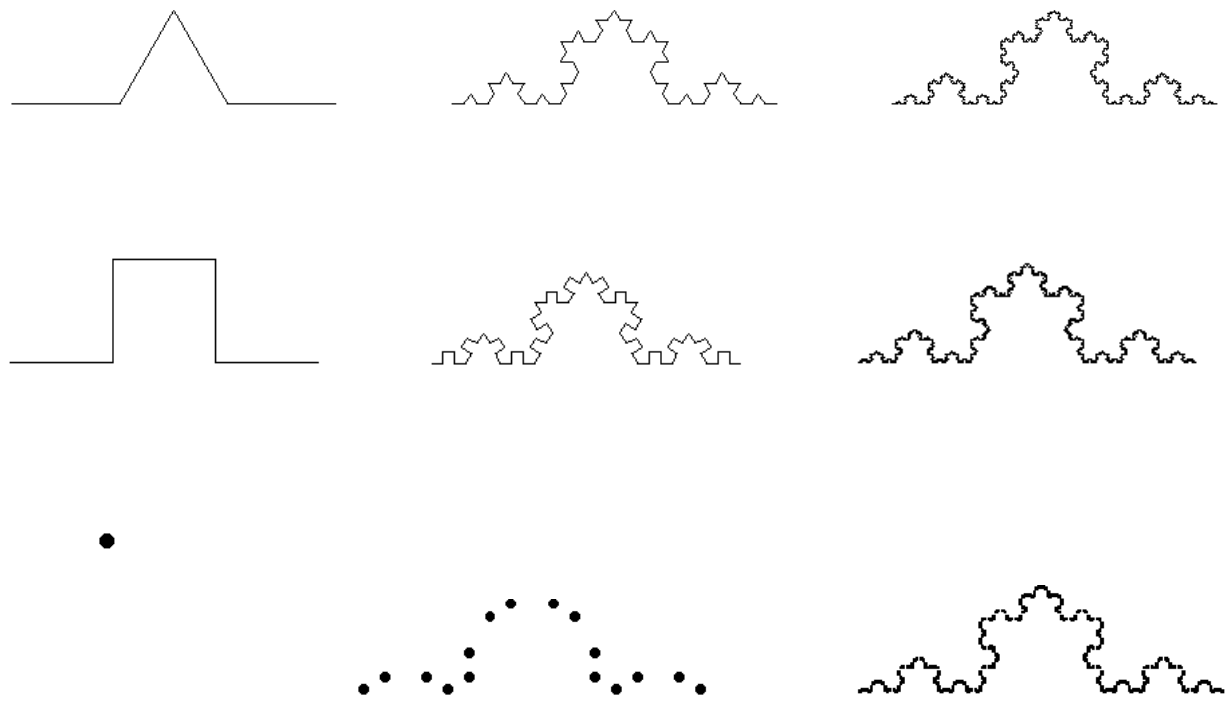


Figure 3: The Koch snowflake -- upper third. The top row is generated starting from a triangular bump; the middle row is generated starting from a square bump instead of a triangular bump; the bottom row is generated from a single point. In each case, levels 0, 2 and 4 of the iteration are illustrated.

Fractals are fixed points of iterated function systems. Thus a fractal is a set S that is mapped onto itself under some collection of contractive maps.

To flesh out these definitions, let us consider once again the Sierpinski triangle and the Koch snowflake. Both of these fractals are self-similar curves -- curves that are generated from scaled down copies of themselves. The Sierpinski triangle is the union of three smaller Sierpinski triangles, and each small bump on the Koch snowflake is a scaled down version of the larger bumps on the snowflake. Self-similarity induces a simple set of affine transformations that map a fractal onto itself. For example, for the Sierpinski triangle S, let $W = \{w_1, w_2, w_3\}$ be the three transformations that scale distances by 1/2 around each of the vertices of the outer triangle. Then $w_1(S), w_2(S), w_3(S)$ are the three small Sierpinski triangles located at each of the vertices of the large Sierpinski triangle, so

$$W(S) = w_1(S) \cup w_2(S) \cup w_3(S) = S .$$

Thus the Sierpinski triangle S is a fixed point of the iterated function system W.

To render the Sierpinski triangle, we can start with the equilateral triangle T whose vertices are located at the outer vertices of the Sierpinski triangle S and iterate the transformation $W = \{w_1, w_2, w_3\}$ on T. That is, we set

$$S_0 = T$$
$$S_1 = W(S_0) = w_1(S_0) \cup w_2(S_0) \cup w_3(S_0)$$
$$\vdots \qquad\qquad \vdots$$
$$S_{n+1} = W(S_n) = w_1(S_n) \cup w_2(S_n) \cup w_3(S_n)$$

In the limit this sequence converges to the fractal triangle -- that is, $S = Lim_{n\to\infty} S_n$ (see Figure 2, top).

Now the key point in this discussion is the following fixed point theorem; we defer the proof of this theorem to the Appendix.

Fixed Point Theorem for Iterated Function Systems

1. Every iterated function system $W = \{w_1, \ldots, w_l\}$ has a unique fixed point A.
2. Let B be a compact set, and let
   $$A_0 = B$$
   $$A_1 = W(A_0) = w_1(A_0) \cup \cdots \cup w_l(A_0)$$
   $$\vdots \qquad\qquad \vdots$$
   $$A_{n+1} = W(A_n) = w_1(A_n) \cup \cdots \cup w_l(A_n) .$$
   Then $A = Lim_{n\to\infty} A_n$.
3. A is independent of B.

Thus a fractal can be rendered by iterating the corresponding iterated function system -- the iterated

function system for which the fractal is a fixed point -- starting with *any* compact set. This convergence is what we mean when we say that the fractal is an attractor: the convergence (attraction) is independent of the compact set on which the iteration begins. This independence from the initial set is the signature of a fractal. In Figures 2 and 3, we illustrate this behavior for the Sierpinski triangle and the Koch snowflake. Notice that in each case the entire fractal can be recovered from a single point.

3. Bezier Subdivision and Iterated Function Systems

Bezier curves are segments of polynomial curves. Each piece of a polynomial curve is just like any other piece of a polynomial curve, so each segment of a Bezier curve is itself a Bezier curve. Thus Bezier curves are self-similar. Therefore, even though Bezier curves are infinitely differentiable, Bezier curves are also fractal curves.

The de Casteljau subdivision algorithm can be used to split a Bezier curve into two Bezier segments. In this section we shall explain how Bezier subdivision is related to iterated function systems, and we shall then apply this fractal interpretation of subdivision to derive a new rendering algorithm for Bezier curves.

To begin, recall that a Bezier curve $P(t)$ of degree n is defined by setting

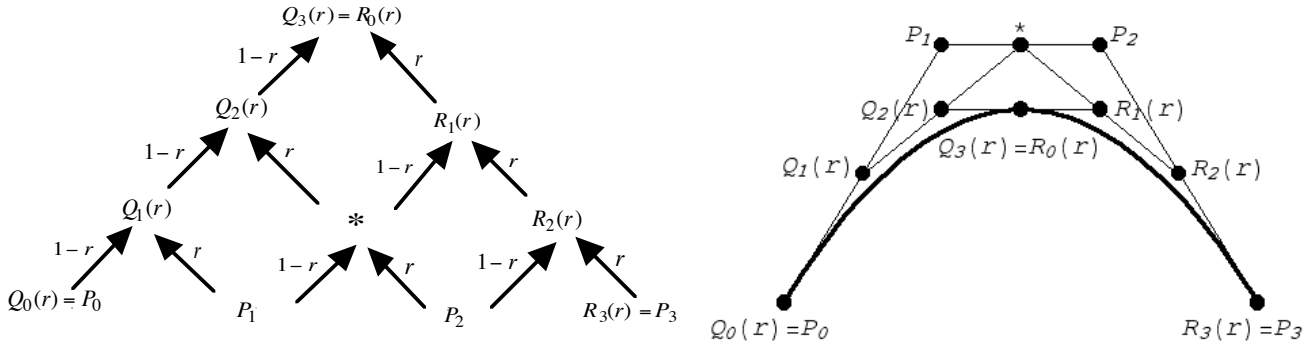
$$P(t) = \sum_{j=0}^{n} B_j^n(t) P_j \qquad 0 \le t \le 1,$$

where $P_0,\ldots,P_n$ are the Bezier control points and

$$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j} \qquad j = 0,\ldots,n$$

are the Bernstein basis functions of degree n. Alternatively, points on a Bezier curve can be evaluated by applying the de Casteljau algorithm (Figure 4) to the control points $P_0,\ldots,P_n$ [2].

A subdivision algorithm is a technique for finding from the original control points $P_0,\ldots,P_n$ new control points $Q_0(r),\ldots,Q_n(r)$ and $R_0(r),\ldots,R_n(r)$ that represent the original Bezier curve restricted to the parameter intervals $[0,r]$ and $[r,1]$ (see Figure 4(b)). The de Casteljau evaluation algorithm is also a subdivision algorithm, since the points that subdivide the Bezier curve emerge along the left and right lateral edges of the de Casteljau triangle [2] (see Figure 4(a)).



(a) Schematic Diagram *(b) Geometric Interpretation*

Figure 4: The de Casteljau algorithm for cubic Bezier curves. (a) A schematic diagram illustrating data flow. The control points are placed at the base of the diagram and the point on the Bezier curve at parameter value r emerges at the apex of the triangle. Each interior node in the diagram is computed by multiplying the values on the arrows that point into the node by the values on the nodes from which the arrows emerge and adding the results. For example, $Q_1(r) = (1-r)P_0 + r\,P_1$. The values $\{Q_k(r)\}$ that emerge along the left lateral edge of the diagram are the control points for the segment of the original Bezier curve in the interval $[0,r]$ and the values $\{R_k(r)\}$ that emerge along the right lateral edge of the diagram are the control points for the segment of the original Bezier curve in the interval $[r,1]$. (b) A geometric interpretation of the de Casteljau algorithm. The points $P_0,\ldots,P_3$ are the control points for the original Bezier curve (dark). The points $Q_0(r),\ldots,Q_3(r)$ are the control points for the same Bezier curve restricted to the interval $[0,r]$, and the points $R_0(r),\ldots,R_3(r)$ are the control points for the same Bezier curve restricted to the interval $[r,1]$.

By construction, each node in the de Casteljau diagram also represents a Bezier curve, albeit of lower degree. Therefore the points that subdivide the original Bezier curve can be computed explicitly from the formulas

$$Q_k(r) = \sum_{j=0}^{k} B_j^k(r) P_j \qquad k = 0,\ldots,n$$

$$R_k(r) = \sum_{j=k}^{n} B_{n-j}^{n-k}(r) P_{k+n-j} \qquad k = 0,\ldots,n.$$

Typically we take $r = 1/2$, and we write $Q_0,\ldots,Q_n$ and $R_0,\ldots,R_n$ in place of $Q_0(r),\ldots,Q_n(r)$ and $R_0(r),\ldots,R_n(r)$.

The explicit formulas for $Q_0,\ldots,Q_n$ and $R_0,\ldots,R_n$ can be represented in matrix form. Let

$$L = \begin{pmatrix} B_0^0(1/2) & 0 & \cdots & 0 \\ B_0^1(1/2) & B_1^1(1/2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ B_0^n(1/2) & B_1^n(1/2) & \cdots & B_n^n(1/2) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \frac{1}{2} & \frac{1}{2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2^n} & \frac{n}{2^n} & \cdots & \frac{1}{2^n} \end{pmatrix} = \left(\frac{\binom{j}{k}}{2^j} \right).$$

Similarly, let

$$M = \begin{pmatrix} B_0^n(1/2) & B_1^n(1/2) & \cdots & B_n^n(1/2) \\ 0 & B_0^{n-1}(1/2) & \cdots & B_{n-1}^{n-1}(1/2) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & B_0^0(1/2) \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{2^n} & \frac{n}{2^n} & \cdots & \frac{1}{2^n} \\ 0 & \frac{1}{2^{n-1}} & \cdots & \frac{1}{2^{n-1}} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} = \left(\frac{\binom{n-j}{n-k}}{2^{n-j}} \right).$$

If $P = (P_0,\ldots,P_n)^T$, then

$$L * P = \begin{pmatrix} B_0^0(1/2) & 0 & \cdots & 0 \\ B_0^1(1/2) & B_1^1(1/2) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ B_0^n(1/2) & B_1^n(1/2) & \cdots & B_n^n(1/2) \end{pmatrix} \begin{pmatrix} P_0 \\ \vdots \\ P_n \end{pmatrix}$$

$$= \begin{pmatrix} Q_0 \\ \vdots \\ Q_n \end{pmatrix}$$

and

$$M * P = \begin{pmatrix} B_0^n(1/2) & B_1^n(1/2) & \cdots & B_n^n(1/2) \\ 0 & B_0^{n-1}(1/2) & \cdots & B_{n-1}^{n-1}(1/2) \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & B_0^0(1/2) \end{pmatrix} \begin{pmatrix} P_0 \\ \vdots \\ P_n \end{pmatrix}$$

$$= \begin{pmatrix} R_0 \\ \vdots \\ R_n \end{pmatrix}.$$

Thus the matrices L, M represent left and right subdivision for Bezier curves.

Starting with the original Bezier control points and applying these matrices repeatedly generates a sequence of control polygons that converge to the original Bezier curve [2,4]. Notice how closely this recursive subdivision algorithm for rendering Bezier curves resembles the iterative rendering algorithm for generating fractals from iterated function systems. Nevertheless, there is one important difference between these two rendering procedures. For fractals we observed that we could start the iteration with any compact set; for Bezier curves we are constrained to start with the Bezier control polygon. This restriction for Bezier curves seems unavoidable because the matrices L, M are independent of the control points $P_0,\ldots,P_n$.

Suppose, however, that the matrix $P = (P_0,\ldots,P_n)^T$ is invertible. Let

$$L_P = P^{-1} * L * P$$

$$M_P = P^{-1} * R * P.$$

Then

$$P * L_P = P * (P^{-1} * L * P) = L * P$$

$$P * M_P = P * (P^{-1} * M * P) = M * P.$$

Moreover, iterating the transformations L_P, M_P -- multiplying now on the right instead of on the left -- generates the same sequence of control polygons as iterating the matrices L, M on the control polygon P. But, and this is the key point, it is easy to show that the matrices L_P, M_P represent contractive maps. Therefore $\{L_P, M_P\}$ is an iterated function system, so we can start with any compact set and the iteration will still converge to the Bezier curve with control points $P_0,\ldots,P_n$. We illustrate this convergence in Figure 5.

For Bezier curves of degree n, the subdivision matrices L, M are $(n+1)\times(n+1)$ matrices. To form the matrices L_P, M_P, the coordinates of the control points $P = (P_0,\ldots,P_n)^T$ must constitute an invertible $(n+1)\times(n+1)$ matrix. If the points $P_0,\ldots P_n$ lie in an n-dimensional affine space, then we can use homogeneous coordinates to form an $(n+1)\times(n+1)$ matrix

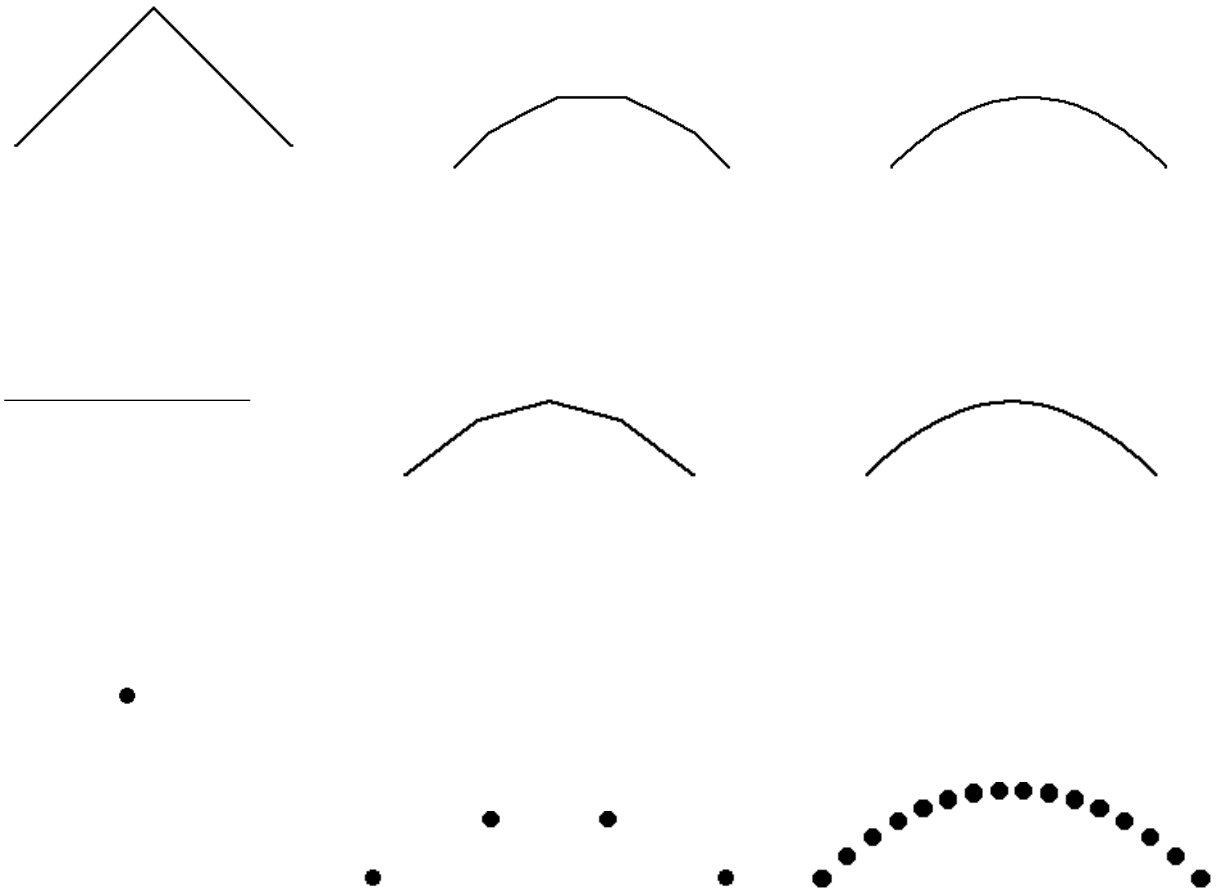


Figure 5: A quadratic Bezier curve generated from an iterated function system. The top row is generated in the usual manner starting from the control polygon; the middle row is generated starting from the chord joining the first and last control points instead of the control polygon; the bottom row is generated from a single point. In each case, levels 0, 2 and 4 of the iteration are illustrated.

$$P = \begin{pmatrix} P_0 & \cdots & P_n \\ 1 & \cdots & 1 \end{pmatrix}^T.$$

The matrix P is then invertible precisely when the points $P_0,\ldots,P_n$ are affinely independent. We used this approach to render the quadratic Bezier curve illustrated in Figure 5.

Typically, however, the degree n of the Bezier curve is larger than the dimension d of the ambient space of the control points. For example, for planar cubic curves, $n=3$ but $d=2$. Nevertheless, in general, we can still form the matrices L_P, M_P by lifting the control points $P=(P_0,\ldots,P_n)^T$ to higher dimensions.

For planar curves, we can proceed in the following fashion. Let (x_k,y_k) be the rectangular coordinates of P_k, $k=0,\ldots,n$. Then we can lift P to higher dimensions by introducing homogeneous coordinates and lifting each control point P_k to dimension k, $3\le k\le n$, by annexing zeroes and ones to the coordinates of P_k as illustrated below:

Quadratics

$$P = \begin{pmatrix} P_0 & 1 \\ P_1 & 1 \\ P_2 & 1 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix}$$

Cubics

$$P = \begin{pmatrix} P_0 & 1 & 0 \\ P_1 & 1 & 0 \\ P_2 & 1 & 0 \\ P_3 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 \\ x_1 & y_1 & 1 & 0 \\ x_2 & y_2 & 1 & 0 \\ x_3 & y_3 & 1 & 1 \end{pmatrix}$$

Quartics

$$P = \begin{pmatrix} P_0 & 1 & 0 & 0 \\ P_1 & 1 & 0 & 0 \\ P_2 & 1 & 0 & 0 \\ P_3 & 1 & 1 & 0 \\ P_4 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 \\ x_1 & y_1 & 1 & 0 & 0 \\ x_2 & y_2 & 1 & 0 & 0 \\ x_3 & y_3 & 1 & 1 & 0 \\ x_4 & y_4 & 1 & 1 & 1 \end{pmatrix}$$

and so on. Notice that in each case, the matrix P is invertible if

$$Det \begin{pmatrix} P_0 & 1 \\ P_1 & 1 \\ P_2 & 1 \end{pmatrix} \neq 0.$$

Thus P is invertible if the points P_0, P_1, P_2 are affinely independent -- that is, if the points P_0, P_1, P_2 are not collinear. Of course, we could choose any other three non collinear control points and this lifting technique would work equally well.

To generate an arbitrary degree n Bezier curve using the iterated function system $\{L_P, M_P\}$, we can now proceed in the following fashion: (i) lift the control points P from the ambient affine space of dimension d to the ambient affine space of dimension n; (ii) generate the corresponding higher dimensional Bezier curve using the iterated function system $\{L_P, M_P\}$; (iii) project the resulting n-dimensional Bezier curve orthogonally back down to the original dimension d. An example of a planar cubic curve generated in this manner is presented in Figure 6.

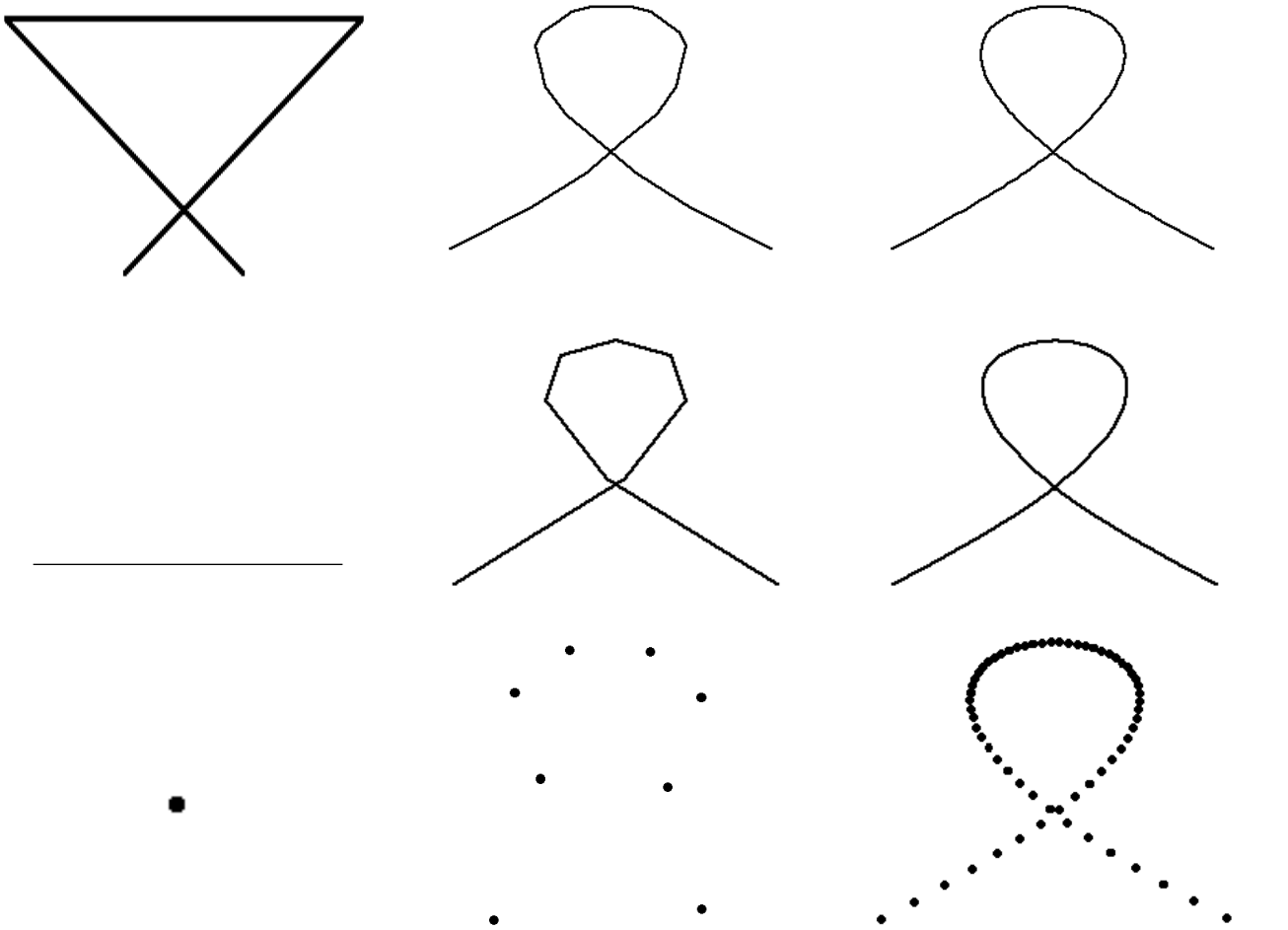


Figure 6: A planar cubic Bezier curve generated from an iterated function system by lifting the control points to 3-dimensions and then projecting the resulting curve back into the plane. The top row is generated in the usual manner starting from the control polygon; the middle row is generated starting from the chord joining the first and last control points instead of the control polygon; the bottom row is generated from a single point. In each case, levels 0, 3 and 6 of the iteration are illustrated.

4. Conclusions

Bezier curves are attractors -- fixed points of iterated function systems derived from the de Casteljau subdivision algorithm. Since both triangular and tensor product Bezier surfaces also have well known subdivision algorithms [2], these Bezier surfaces are also attractors. Therefore three sided and four sided Bezier surfaces are fractal surfaces which can be generated from iterated function systems.

Multisided Bezier patches, such as S-patches [2,5] and toric Bezier patches [2,3] are also studied in Geometric Modeling. It seems reasonable to expect that these multisided surfaces are also attractors, but the details of how to generate the corresponding iterated function systems are still an open question.

Acknowledgment

I would like to thank Joe Warren for first pointing out to me the connection between Bezier subdivision and iterated function systems.

References

[1] Barnsley, M. (1993), *Fractals Everywhere*, (Second Edition), Academic Press, Boston, Mass.

[2] Goldman, R. (2002), *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*, Morgan Kaufmann, San Francisco.

[3] Krasauskas, R. (2000), Toric surface patches, *Advances in Computational Mathematics*, Vol. 21, pp. 1-25.

[4] Lane, J. and Riesenfeld, R (1980), A theoretical development for the computer generation and display of piecewise polynomial surfaces, *IEEE Trans. on Pattern Anal. and Mach. Intell.,* Vol. 2, pp. 35-46.

[5] Loop, C.T. and DeRose, T.D. (1989), A multisided generalization of Bezier surfaces, *TOG,* Vol. 8, pp. 204-234.

Appendix: The Trivial Fixed Point Theorem and its Consequences

The signature property of attractors -- that convergence is independent of the starting set -- is a consequence of the following theorem.

The Trivial Fixed Point Theorem:
Suppose that

i. *T is a contractive map on a complete metric space;*

ii. $P_{n+1} = T(P_n)$ *for all* $n \geq 0$.

Then $Lim_{n\to\infty} P_n$ *exists and is the unique fixed point of T for any choice of* P_0*!*

The purpose of this Appendix is to derive this fixed point theorem and then use this theorem to establish the signature property of attractors. We begin with a series of simple lemmas. For further details and additional proofs, see [1].

Lemma 1: *Suppose that*

i. *T is a contractive map;*

ii. $P_{n+1} = T(P_n)$ *for all* $n \geq 0$.

If $P = Lim_{n\to\infty} P_n$*, then P is a fixed point of T.*

Proof: Since T is contractive, T is continuous. Therefore,

$$T(P) = T\left(Lim_{n\to\infty} P_n\right) = Lim_{n\to\infty} T(P_n) = Lim_{n\to\infty} P_{n+1} = P.$$

Lemma 2: *Suppose that T is a contractive map. Then T has at most one fixed point.*

Proof: If P and Q are both fixed points of T, then $T(P) = P$ and $T(Q) = Q$. Therefore,

$$Dist\left(T(P), T(Q)\right) = Dist(P, Q).$$

Hence, contrary to assumption, T is not contractive. Therefore T can have at most one fixed point.

Before we can state our next lemma, we need to recall the notion of a cauchy sequence. A sequence $\{P_n\}$ is *cauchy* if for every $\varepsilon > 0$ there is an integer N such that $Dist(P_{n+m}, P_n) < \varepsilon$ for all $n > N$. Intuitively, a sequence of points $\{P_n\}$ is *cauchy* if the points get closer and closer as n gets larger and larger.

Lemma 3: *Suppose that*

i. *T is a contractive map;*

ii. $P_{n+1} = T(P_n)$ *for all* $n \geq 0$.

Then $\{P_n\}$ *is a cauchy sequence for any choice of* P_0.

Proof: Since T is a contractive map,

$$\begin{aligned} Dist\left(P_{n+1}, P_n\right) &= Dist\left(T(P_n), T(P_{n-1})\right) \\ &\leq s\,Dist\left(P_n, P_{n-1}\right) = s\,Dist\left(T(P_{n-1}), T(P_{n-2})\right) \\ &\vdots \\ &\leq s^n Dist\left(P_1, P_0\right). \end{aligned}$$

Therefore, for n sufficiently large,

$$\begin{aligned} Dist\left(P_{n+m}, P_n\right) &\leq Dist\left(P_{n+m}, P_{n+m-1}\right) + \cdots + Dist\left(P_{n+1}, P_n\right) \\ &\leq (s^{n+m-1} + \cdots + s^n) Dist\left(P_1, P_0\right) \\ &\leq s^n \frac{Dist\left(P_1, P_0\right)}{1-s} \\ &< \varepsilon. \end{aligned}$$

With these results in hand, we are almost ready to prove the trivial fixed point theorem. But before we proceed, we need to explain what we mean by a *complete metric space*.

A *metric space* (X, d) is a space X together with a

function d that measures the distance between any two elements in the space X. The function d must satisfy the usual properties of distance; in particular, d must satisfy the triangular inequality:

$$d(x,z) \le d(x,y) + d(y,z).$$

A metric space is said to be *complete* if every cauchy sequence converges. The metric space $(\mathbf{R^n}, dist)$, where $dist$ is the standard Euclidean metric on $\mathbf{R^n}$, is a complete metric space.

The Trivial Fixed Point Theorem:

Suppose that

i. T is a contractive map on a complete metric space;

ii. $P_{n+1} = T(P_n)$ *for all* $n \ge 0$.

Then $Lim_{n\to\infty} P_n$ *exists and is the unique fixed point of T for any choice of* P_0*!*

Proof: This result is a consequence of the following observations:

i. the sequence $\{P_n\}$ is cauchy -- Lemma 3;

ii. $Lim_{n\to\infty} P_n$ exists -- since, by assumption, the metric space is complete, so every cauchy sequence converges;

iii. $Lim_{n\to\infty} P_n$ is a fixed point of T -- Lemma 1;

iv. the fixed point of T is unique -- Lemma 2.

The trivial fixed point theorem can be used to establish the signature property of attractors -- that convergence is independent of the starting set -- by introducing a new metric space consisting of the compact subsets of $\mathbf{R^n}$.

Let $H(\mathbf{R^n})$ denote the space of compact subsets of $\mathbf{R^n}$. (A subset of $\mathbf{R^n}$ is *compact* if it lies within a sphere of finite radius and contains its own boundary.) We can measure the distance between any two compact sets R,S in $H(\mathbf{R^n})$ by adopting the Hausdorff metric $h(R,S)$.

To construct the Hausdorff metric, let x be an arbitrary point in $\mathbf{R^n}$, and define

$$Dist(x,S) = Min_{y\in S}\{dist(x,y)\}$$
$$Dist(R,S) = Max_{x\in R}\{Dist(x,S)\}$$
$$h(R,S) = Max\{Dist(R,S), Dist(S,R)\}.$$

Then h is a metric on $H(\mathbf{R^n})$. Moreover, the following results are established in [1].

Theorem 1: *The space* $(H(\mathbf{R^n}), h)$ *is a complete metric space.*

Theorem 2: *Let* $W = \{w_1,\ldots,w_l\}$ *be an iterated function system -- i.e. W consists of a finite collection of contractive maps* $w_1,\ldots,w_l$. *Then W is a contractive map on* $H(\mathbf{R^n})$. *That is, if the maps* $w_1,\ldots,w_l$ *all shrink the distance between points, then the map W shrinks the distance between compact sets.*

Now the trivial fixed point theorem combined with Theorems 1 and 2 immediately implies the following signature property of attractors.

Fixed Point Theorem for Iterated Function Systems

1. Every iterated function system $W = \{w_1,\ldots,w_l\}$ *has a unique fixed point A.*

2. Let B be a compact set, and let

$$A_0 = B$$
$$A_1 = W(A_0) = w_1(A_0) \cup \cdots \cup w_l(A_0)$$
$$\vdots \qquad\qquad \vdots$$
$$A_{n+1} = W(A_n) = w_1(A_n) \cup \cdots \cup w_l(A_n).$$

Then $A = Lim_{n\to\infty} A_n$.

3. A is independent of B.