Curves as fractals

Krzysztof Gdawiec

kgdawiec@ux2.math.us.edu.pl

1 Introduction

Usually we associate the fractals with Kochs curve, Sierpinski's gasket or Cantors set. All of them are some strange creatures. It turns out that also Bézier curves are fractals what R. Goldman has shown in [5]. This is an astounding fact because we look on Bézier curves as something regular and something that do not posses strange properties as the fractals mentioned earlier. Goldman in [14] went considerably further and has shown that any curve and surface which we can achive with help of subdivision scheme is also a fractal. Moreover in [8] authors have shown how to construct IFS for linear segment what we could not achive with method presented by Goldman. Because we can split any contour into finite number of curves and linear segments and then find IFS for these curves and linear segments we are able to generate any contour in a fractal way.

In this paper we present IFSs for Bézier curves, linear segment and for Chaikin's curves. Next we show some shapes generated in a fractal way.

2 Fractals as attractors

Let us remind some basic facts about fractals which we will need in our further work. Lets begin from the definition of contraction mapping. We say that mapping $f : X \to X$ ((X, ρ) – metric space) is a contraction mapping if there exists $0 \leq c < 1$ such that

$$\forall_{x,y \in X} \quad \rho(f(x), f(y)) \leq c\rho(x, y)$$

Let $f : X \to X$ be contraction mapping. Then we can use mapping f to any subset $A \subset X$ as follows

$$f(A) = \{f(a) : a \in A\}$$

The set $W = \{f_1, \ldots, f_n\}$, where $n \in \mathbb{N}$ and $f_i : X \to X$ ((X, ρ) – complete metric space) are contraction mappings for $i = 1, \ldots, n$ we call Iterated Function System (IFS). As in the case of contraction mapping IFS defines us mapping for any subset $A \subset X$ as follows

$$W(A) = \bigcup_{i=1}^{n} f_i(A)$$

Now for our IFS let us consider following sequence

$$\begin{cases} W^0(A) = A \\ W^n(A) = W(W^{n-1}(A)), \quad n \geq 1 \end{cases}$$

Then the limit of this sequence we will call the attractor of mapping W. And this attractor we will call fractal generated by IFS W.

Like in [4] as the contraction mappings we will be using affine mappings $f : \mathbb{R}^2 \to \mathbb{R}^2$ given by equation

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

where $a, b, c, d, e, f \in \mathbb{R}$, which we can write in homogenous coordinates as follows

$$f\left([x, y, 1]\right) = [x, y, 1] \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ e & f & 1 \end{bmatrix} = [x, y, 1]F$$

where

$$F = \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ e & f & 1 \end{bmatrix}$$

To generate fractals we will use two algorithms: IFS and IFS with probabilities (Chaos game). Description of these algorithms we can find in [4].

3 Bézier curves as fractals

In this section we will present IFS for Bézier curves, but first we remind what is a Bézier curve.

Bézier curve $P(t)$ of degree $n \in \mathbb{N}$ is parametric curve defined as follows

$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t), \qquad t \in [0, 1]$$

where $P_0, \ldots, P_n \in \mathbb{R}^2$ are the control points and

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

are the Bernstein polynomials.

Examples of Bézier curves of degree 2 and 3 we can see in the fig.1.

With help of de Casteljau algorithm we can find point on Bézier curve for a given parameter $t \in [0, 1]$. Moreover it gives us information about the division of this curve into two parts from which every one is a Bézier curve. We can calculate the control points of these curves with help of following equations

$$Q_k(t) = \sum_{i=0}^{k} P_i B_i^k(t)$$

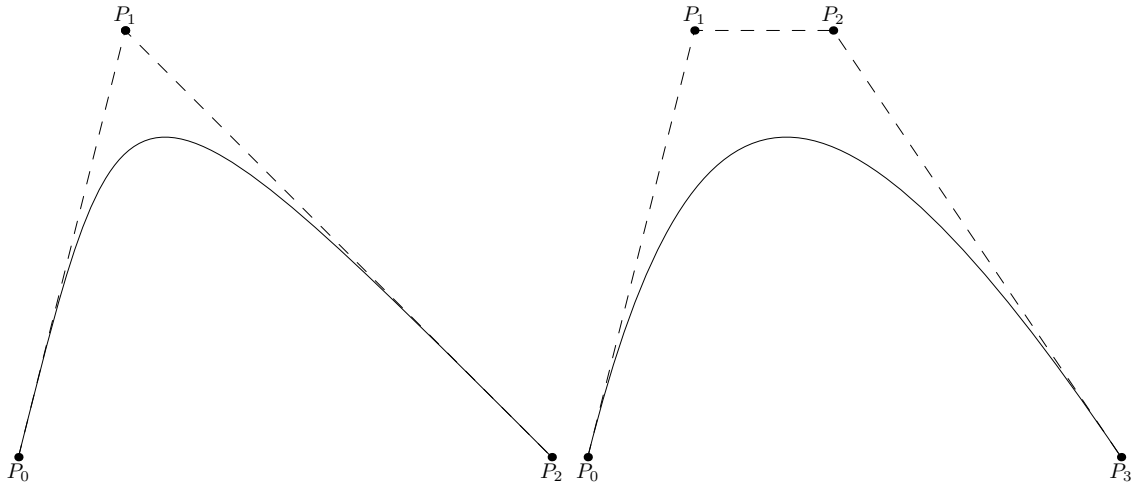

Fig. 1: Bézier curves of degree 2 and 3

$$R_k(t) = \sum_{i=k}^{n} P_{k+n-i} B_{n-i}^{n-k}(t)$$

for $k = 0, \ldots, n$ and $t \in [0, 1]$.

Figure 2 illustrates the use of de Casteljau algorithm for Bézier curve of degree 3 and parameter $t = \frac{1}{2}$.

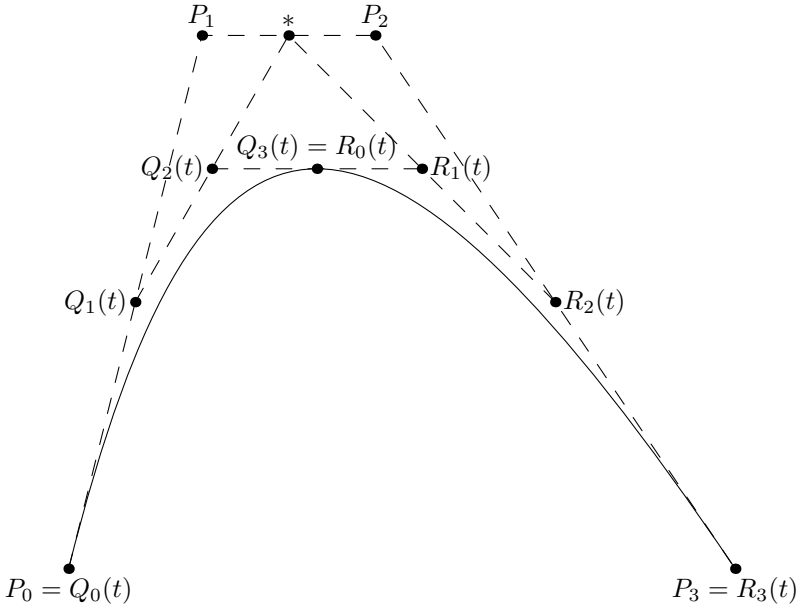

Fig. 2: de Casteljau algorithm

Let us introduce following martices which will let us denote equations for $Q_0, \ldots, Q_n$,

$R_0, \ldots, R_n$ in matrix form

$$L = \begin{bmatrix} B_0^0(t) & 0 & 0 & \ldots & 0 \\ B_0^1(t) & B_1^1(t) & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ B_0^n(t) & B_1^n(t) & B_2^n(t) & \ldots & B_n^n(t) \end{bmatrix}$$

$$M = \begin{bmatrix} B_0^n(t) & B_1^n(t) & \ldots & B_{n-1}^n(t) & B_n^n(t) \\ 0 & B_0^{n-1}(t) & \ldots & B_{n-2}^{n-1}(t) & B_{n-1}^{n-1}(t) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & B_0^0(t) \end{bmatrix}$$

Then $L \cdot P = [Q_0 \cdots Q_n]^T$ and $M \cdot P = [R_0 \cdots R_n]^T$ where $P = [P_0 \cdots P_n]^T$.

If matrix P is not a square matrix, then we should embed it in the higher dimensional space, method can be found in [5]. Let us supose that our matrix P is invertible, that is control points are not colinear. Then the form of IFS for our Bézier curve is following $IFS = \{P^{-1} \cdot L \cdot P, P^{-1} \cdot M \cdot P\}$.

For our further purposes we will be only consider Bézier curves of degree 1 and 2 and parameter $t = \frac{1}{2}$. Matrices L, M, P for the curve of degree 2 and $t = \frac{1}{2}$ are following

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix}, \quad M = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}, \quad P = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}$$

where $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ are the co-ordinates of the control points of the Bézier curve.

In the case of curves of degree 1 (linear segments) it is not possible to embed matrix P in the higher dimensional space so, that it is invertible. So we are not able to achive IFS in the way described above. The IFS for linear segment where $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ are the ends of this linear segment, looks as follows $IFS = \{F_1, F_2\}$ where

$$F_1 = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ \frac{x_0}{2} & \frac{y_0}{2} & 1 \end{bmatrix}, \quad F_2 = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ \frac{x_1}{2} & \frac{y_1}{2} & 1 \end{bmatrix}$$

4 Chaikin curves as fractals

Bézier curves introduced in previous section were based on analitical representation. In 1976 George Chaikin presented completly new method for generating curves. It was purely geometric and based on ,,corner cutting". For any control polygon $P_0, \ldots, P_n$ ($n \in \mathbb{N}$) we create new polygon $Q_0, R_0, Q_1, R_1, \ldots, Q_{n-1}, R_{n-1}$ where

$$Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1}$$

$$R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

for $i = 0, \ldots, n-1$. Next we repeat that procedure for this new polygon. After few iterations we achive a smooth curve. Figure 3 presents the idea of this algorithm.

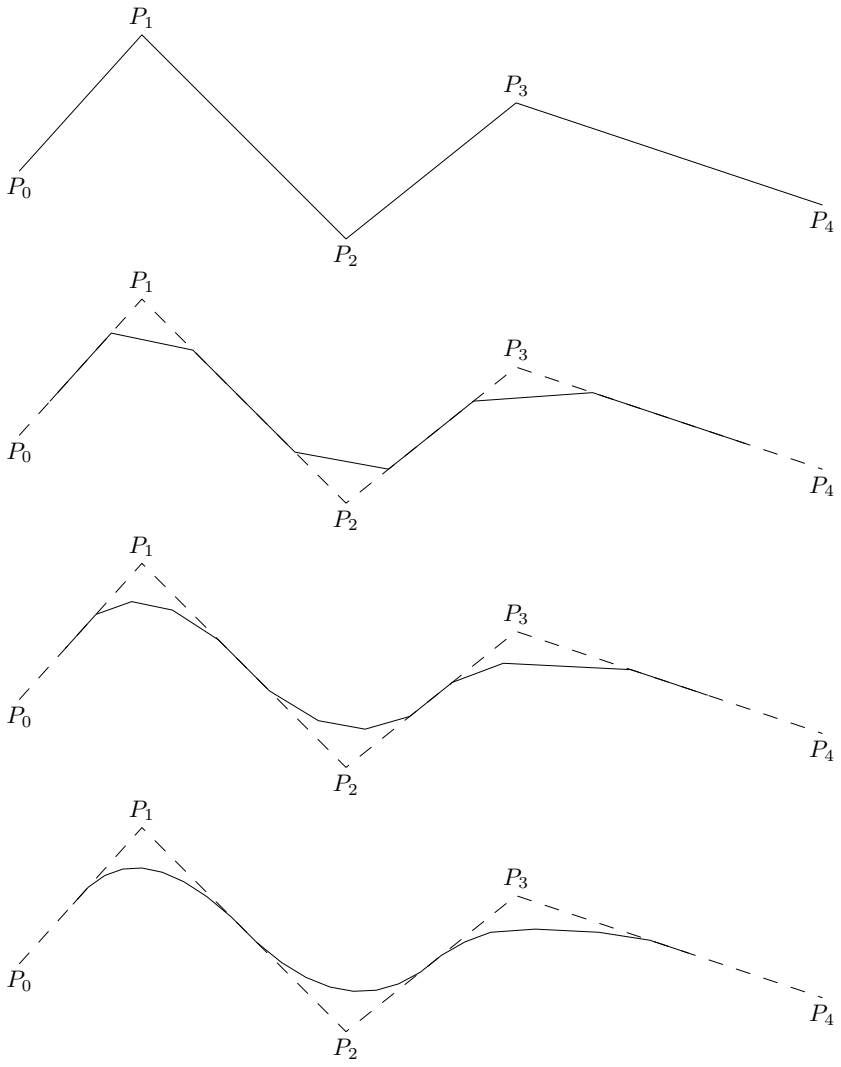

Fig. 3: Chaikin's algorithm steps $0, 1, 2, 3$

It turns out that Chaikin's curve is equivalent to the quadratic uniform B-spline, which is equivalent to the picewise quadratic Bézier curve.

Similary as in the case of Bézier curve and de Casteljau algorithm also for Chaikin's curve we can find IFS generating given curve. The form of this IFS looks as follows $IFS = \{P^{-1} \cdot L \cdot P, P^{-1} \cdot M \cdot P\}$ where

$$L = \begin{bmatrix} \frac{3}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} \end{bmatrix}, \quad M = \begin{bmatrix} \frac{1}{4} & \frac{3}{4} & 0 \\ 0 & \frac{3}{4} & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{3}{4} \end{bmatrix}, \quad P = \begin{bmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{bmatrix}$$

In distinction from Bézier curves points P_0 and P_2 in Chaikin's curve do not interpolate ends of this curve. The ends lie in the half of the distance betwen P_0, P_1 and P_1, P_2. Figure 4 ilustrates this situation.

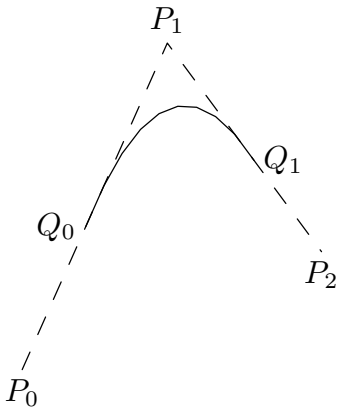

Fig. 4: Chaikin's curve

Similary as in the case of Bézier curves we can not generate linear segment with help of the IFS mentioned above. Also in this case we can rid of this inconvenience by intorducing seperate IFS for linear segment which looks as follows $IFS = \{P^{-1} \cdot L_s \cdot P, P^{-1} \cdot M_s \cdot P\}$ where

$$L_s = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \quad M_s = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and matrix P is defined such as for the typical Chaikin curve. Figure 5 presents linear segment achived by Chaikin's algorithm with equation

$$Q_i = R_i = \frac{1}{2}P_i + \frac{1}{2}P_{i+1}$$

for $i = 0, \ldots, n-1$.

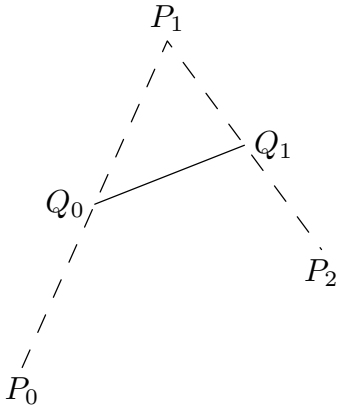

Fig. 5: Chaikin's linear segment

5 Program and examples

In this section we will talk over the program for generating curves in a fractal way and also we present some images achived with help of our program.

With our program it is possible to generate Bézier curves and Chaikin's curves in a fractal way, which we have talked about in previous sections. Lets begin with the input data for our program. As the input data we pass control points of the curves. For this purpose we use text files with extensions *.bez* for Bézier curves and *.ch* for Chaikin's curves. The format of these files is following. The first line contains a number $n \in \mathbb{N}$ of the segments. The next n lines describe these segments. First character in the line marks if this is a linear segment S or curve B in the Bézier curve case, C in the Chaikin's curve case, next we give the coordinates of the control points.

Listing 1: Example of *.bez* file

```
3
B  0  10  10  −10  20  10
S  0  10  10  20
S  20  10  10  20
```

The listing 1 presents an example file with data for the Bézier case, which defines Bézier curve with control points $P_0 = (0, 10)$, $P_1 = (10, -10)$, $P_2 = (20, 10)$ and two linear segments with ends $R_0 = (0, 10)$, $R_1 = (10, 20)$ for the first and $T_0 = (20, 10)$, $T_1 = (10, 20)$ for the second.

Listing 2: Example of *.ch* file

```
3
C  0  10  10  −10  20  10
S  0  10  5  10  10  20
S  20  10  15  10  10  20
```

The listing 2 presents an example file with data for the Chaikin case, which like in the Bézier case defines one Chaikin's curve with control points $P_0 = (0, 10)$, $P_1 = (10, -10)$, $P_2 = (20, 10)$ and two linear segments with control points $R_0 = (0, 10)$, $R_1 = (5, 10)$, $R_2 = (10, 20)$ for the first one and $T_0 = (20, 10)$, $T_0 = (15, 10)$, $T_1 = (10, 20)$ for the second.

For the fractal generation we have used the same methods as in the [4].

Figures 6-11 present examples for the Bézier case and figures 12-16 present examples for the Chaikin case.

References

[1] G. Chaikin, *An algorithm for high speed curve generation*, Computer Graphics and Image Processing, 3, (1974), 346-349

[2] M. Barnsley, *Fractals Everywhere*, Academic Press, Boston, (1993)

[3] Y. Fisher, *Fractal image compression*, SIGGRAPH '92 Course Notes, vol. 12 pp. 7.1-7.19, (1992)

[4] K. Gdawiec, *Fractals*, MathWorks, (2006)
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10919

[5] R. Goldman, *The fractal nature of Bézier curves*, Proceedings of the Geometric Modeling and Processing, April 13-15, Beijing, China, (2004), 3-11

[6] John C. Hart, *Fractal Image Compression and Recurent Iterated Function Stystems*, IEEE Computer Graphics and Applications, vol. 16, no. 4, pp. 25-33, (1996)

[7] P. Koprowski, *Okruchy geometrii komputerowej*,
http://unix.math2.us.edu.pl/~perry/gk/okruchy.pdf

[8] W. Kotarski, A. Lisowska, *On Fractal Modeling of Contours*, Maplesoft, (2005),
http://www.maplesoft.com/applications/app_center_view.aspx?AID=1651

[9] W. Kotarski, A. Lisowska, *Probabilistic Approach to Fractal Modeling of Shapes*, Maplesoft, (2005),
http://www.maplesoft.com/applications/app_center_view.aspx?AID=1657

[10] W. Kotarski, A. Lisowska, *On Geometric Chaikin's Approach to Fractal Modeling of Contours*, Maplesoft, (2005),
http://www.maplesoft.com/applications/app_center_view.aspx?AID=1667

[11] W. Kotarski, A. Lisowska, *On Bézier-Fractal Modeling of 2D Shapes*, International Journal of Pure and Applied Mathematics, (2005), vol. 24, no. 1, 123-134

[12] G. Neil, K. M. Curtis, *Shape recognition using fractal geometry*, Pattern Recognition, vol. 30, No. 12, pp. 1957-1969, (1997)

[13] H.-O. Peitgen, H. Jurgens, D. Saupe, *Fractals for the Classroom*, Springer-Verlag, (1992)

[14] S. Schaefer, D. Levin, R. Goldman, *Subdivision Schemes and Attractors*, Symposium on Geometry Processing, (2005), 171-180

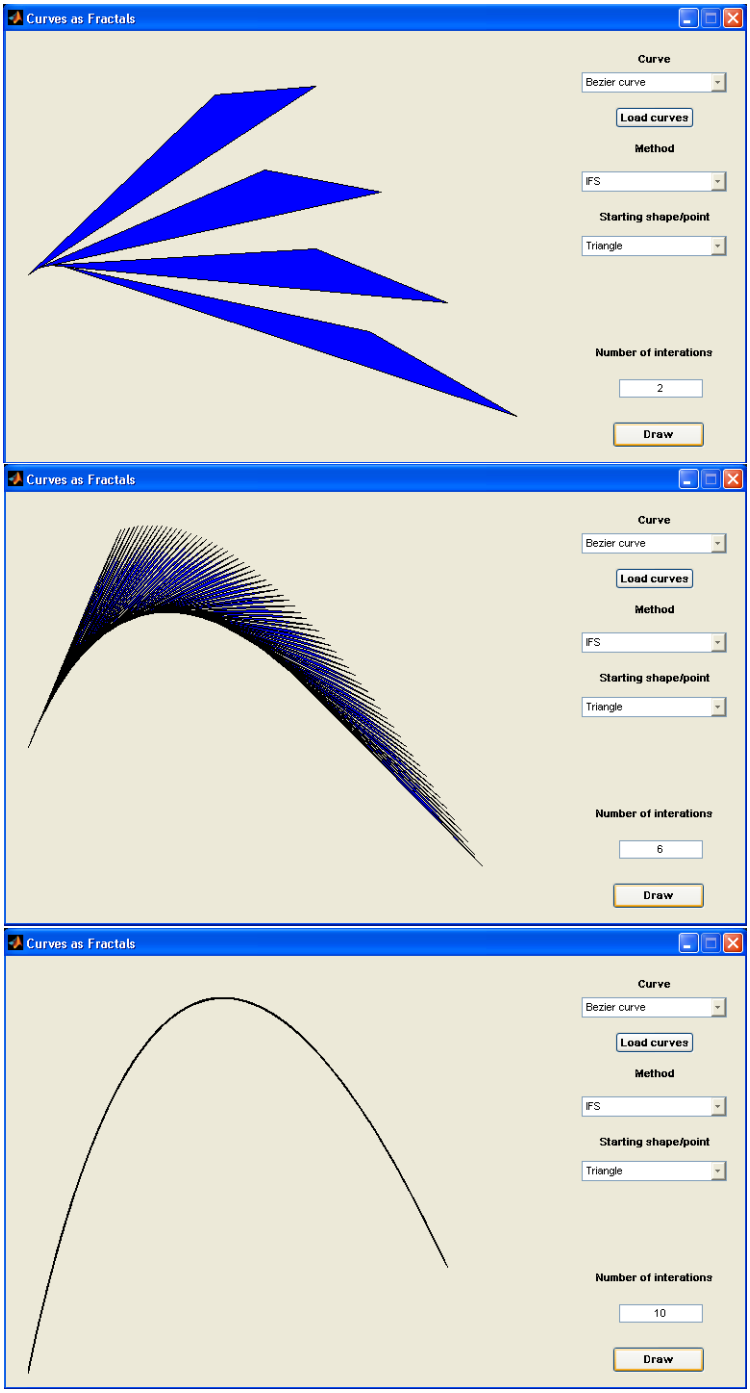

Fig. 6: Single Bézier curve – starts from triangle, iterations 2, 6, 10

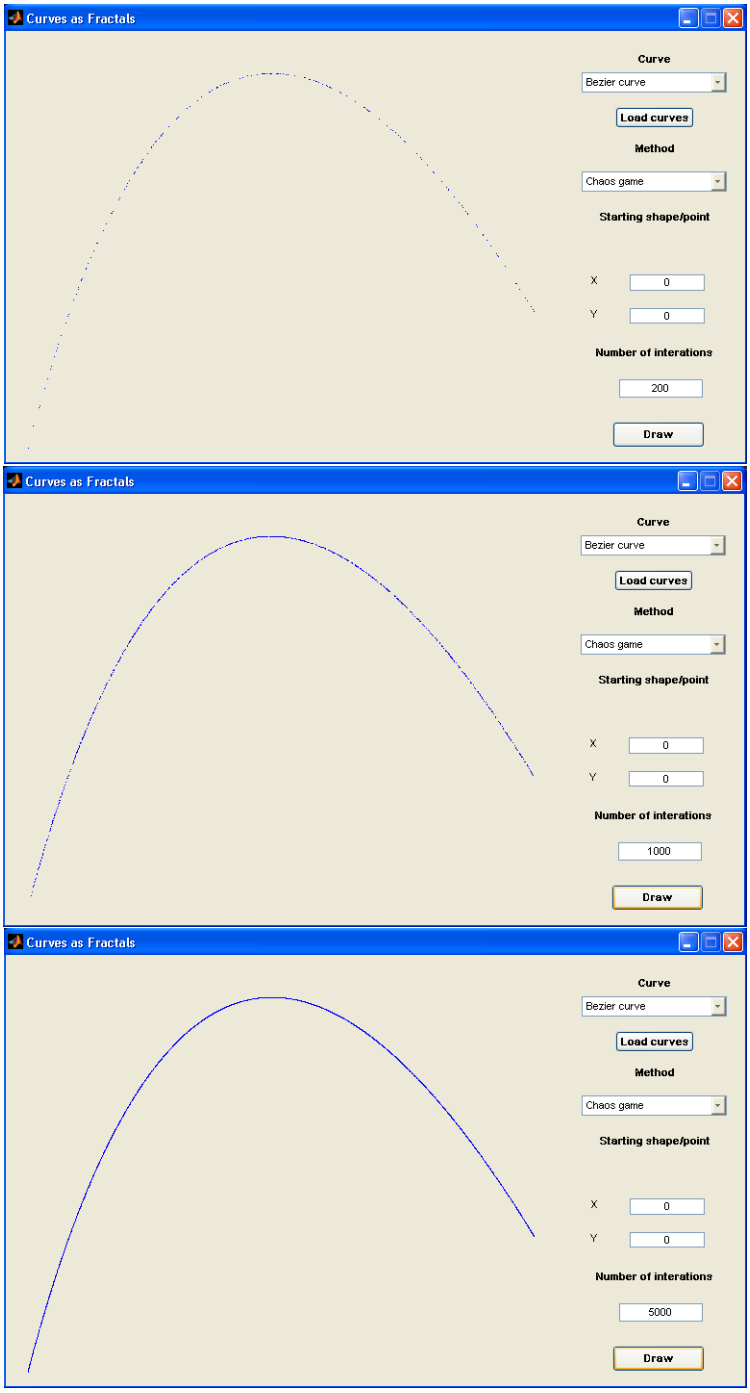

Fig. 7: Single Bézier curve – Chaos game iterations 200, 1000, 5000

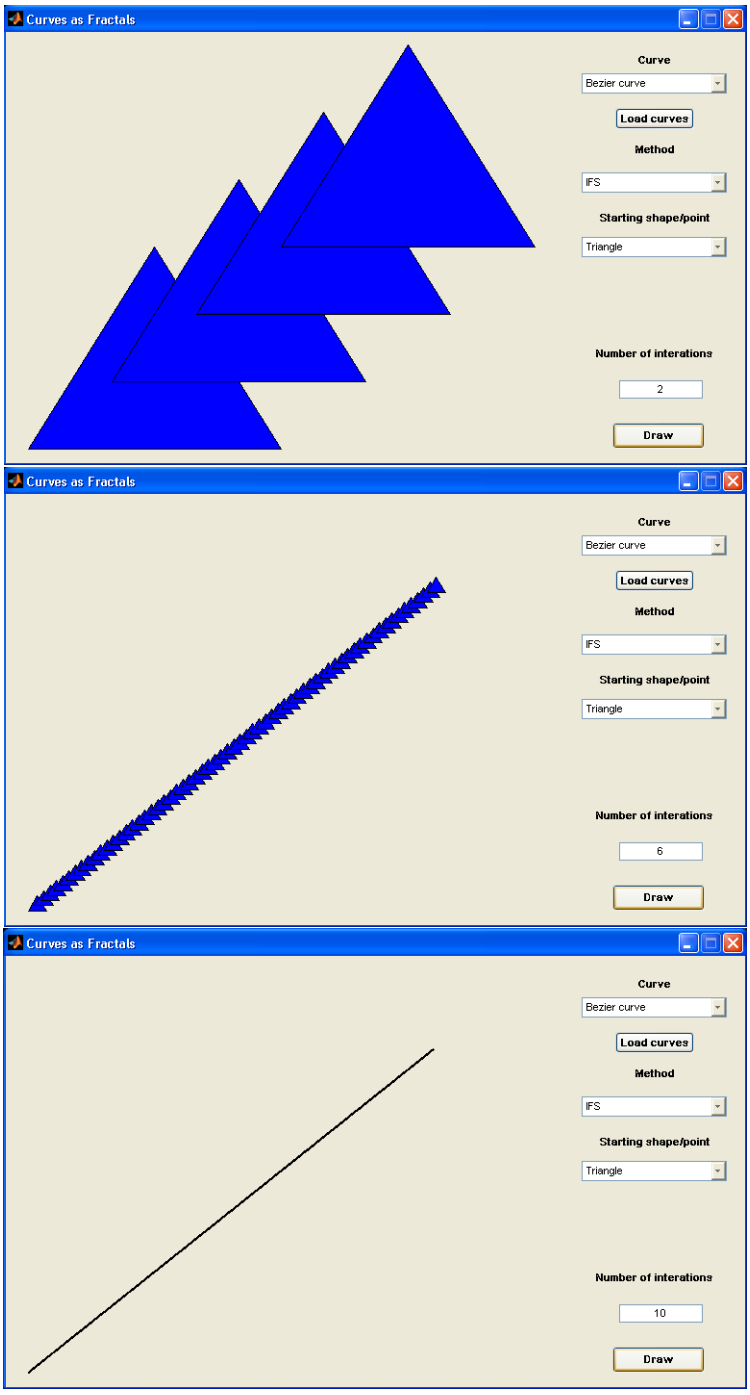

Fig. 8: Single linear segment (Bézier case) – starts from triangle, iterations 2, 6, 10

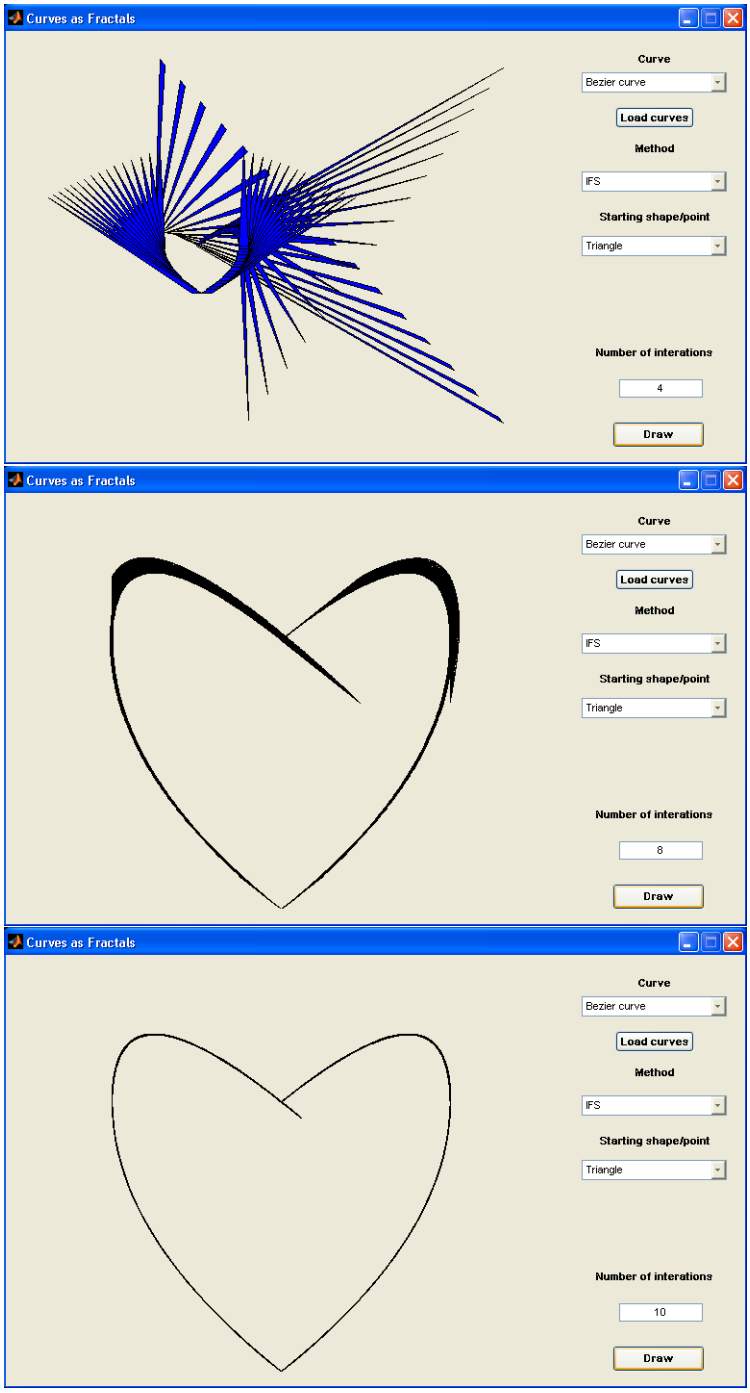

Fig. 9: Heart – starts from triangle, iterations 4, 8, 10

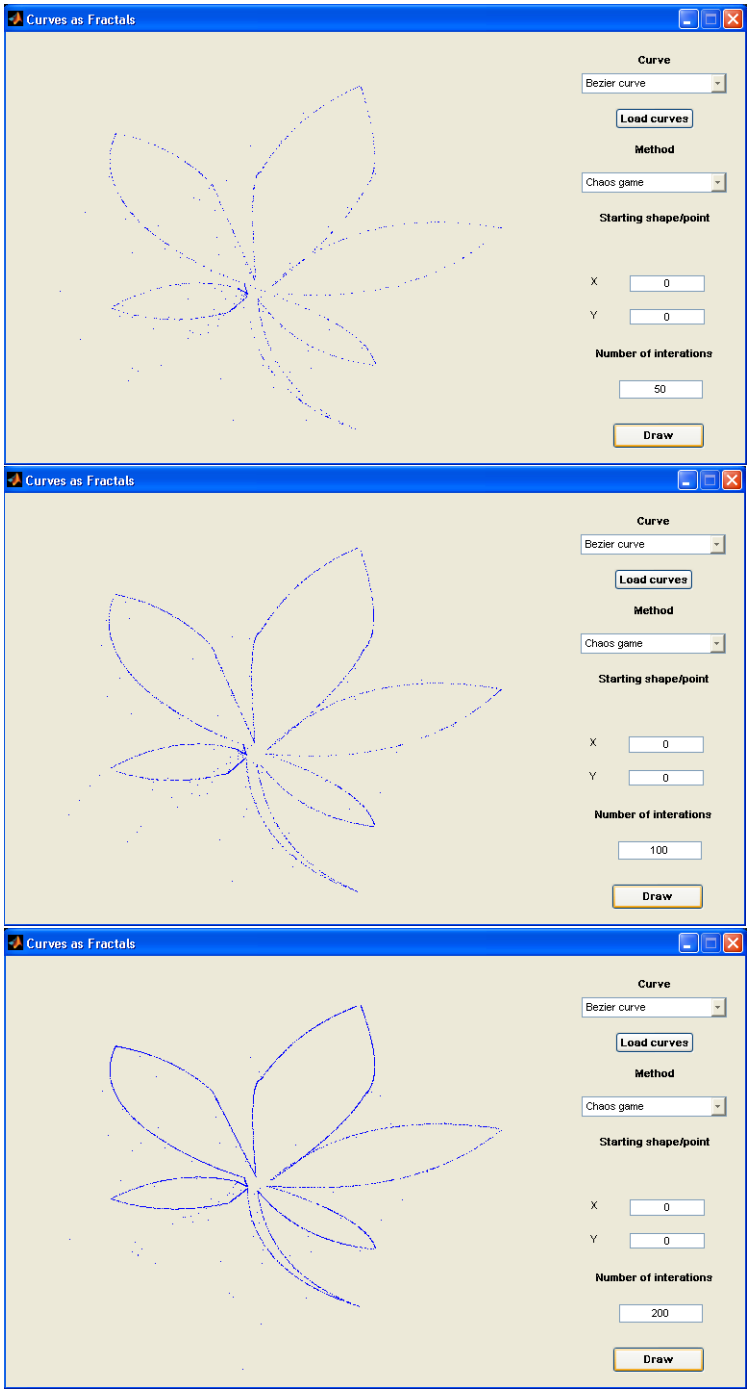

Fig. 10: Ohio buckeye – Chaos game iterations 50, 100, 200

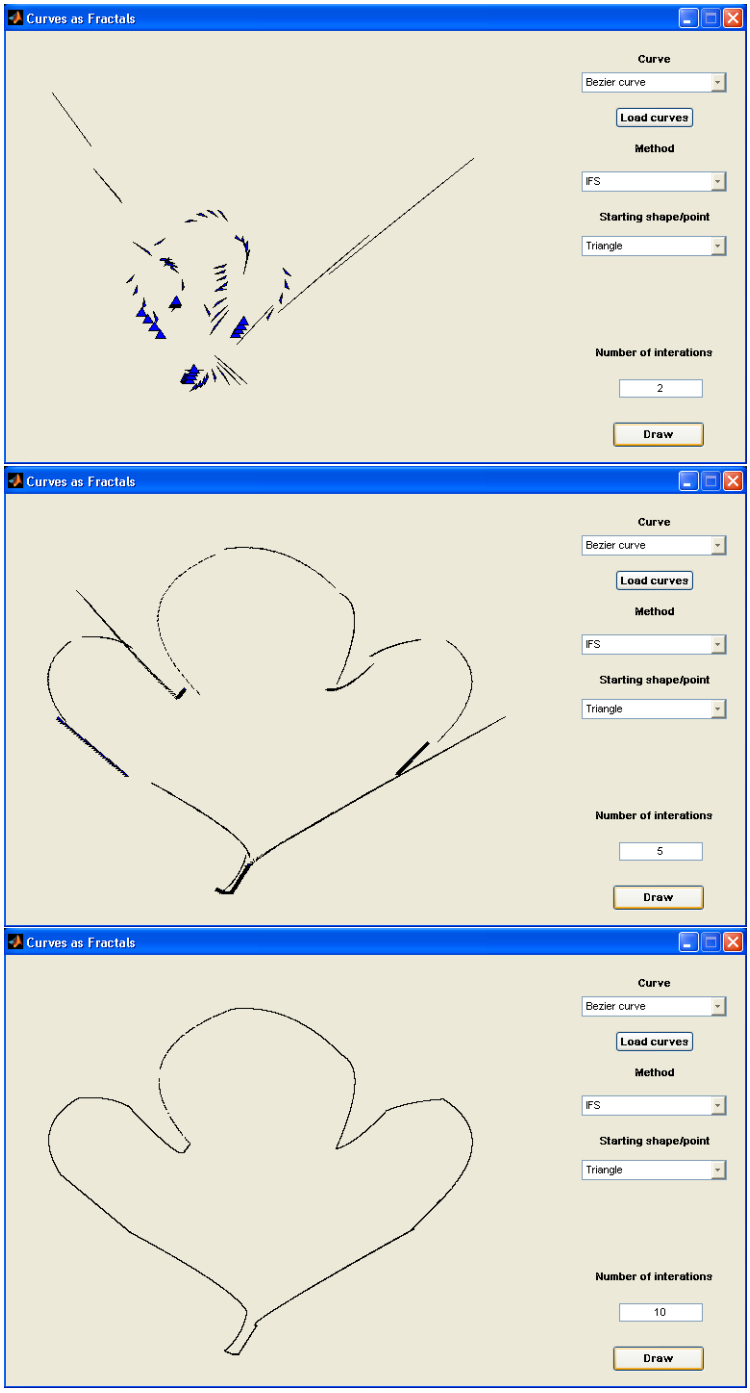

Fig. 11: Sassafras double – starts from triangle, iterations 2, 5, 10

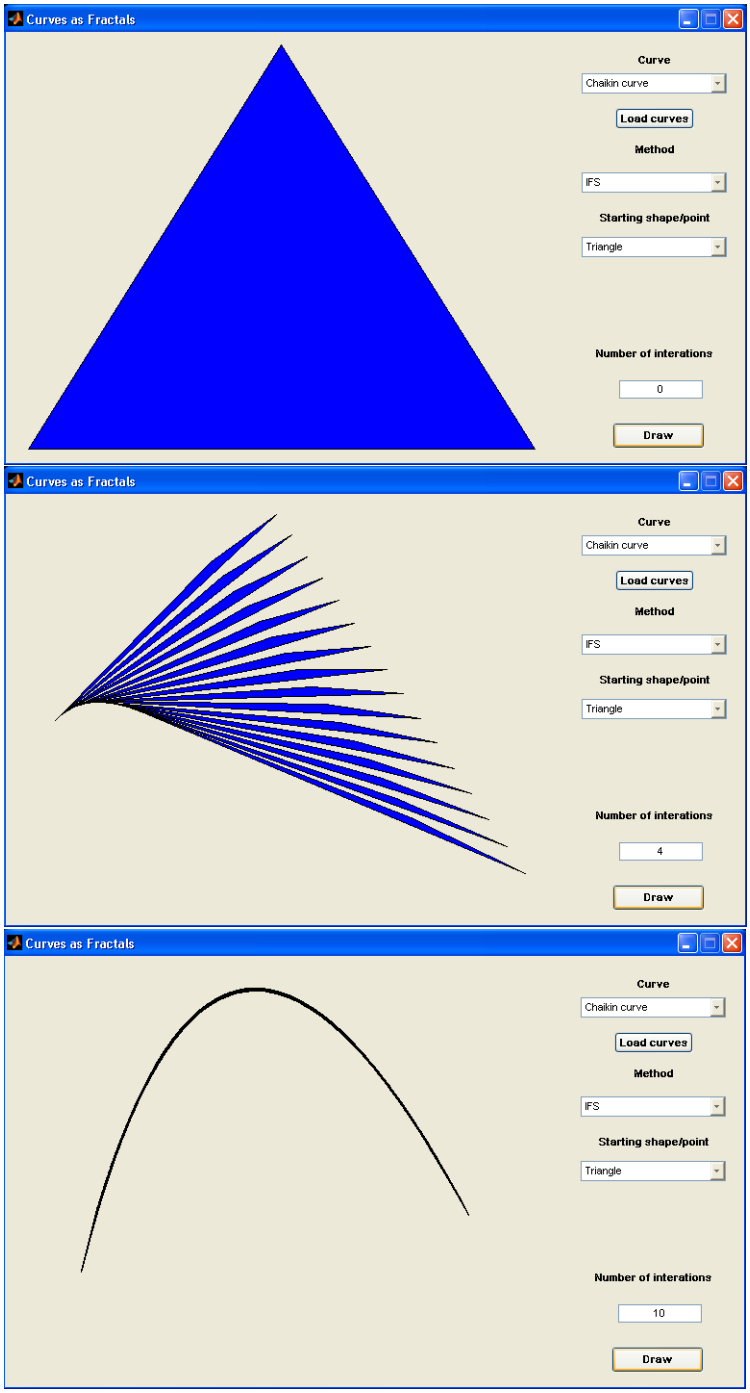

Fig. 12: Single Chaikin's curve – starts from triangle, iterations 0, 4, 10

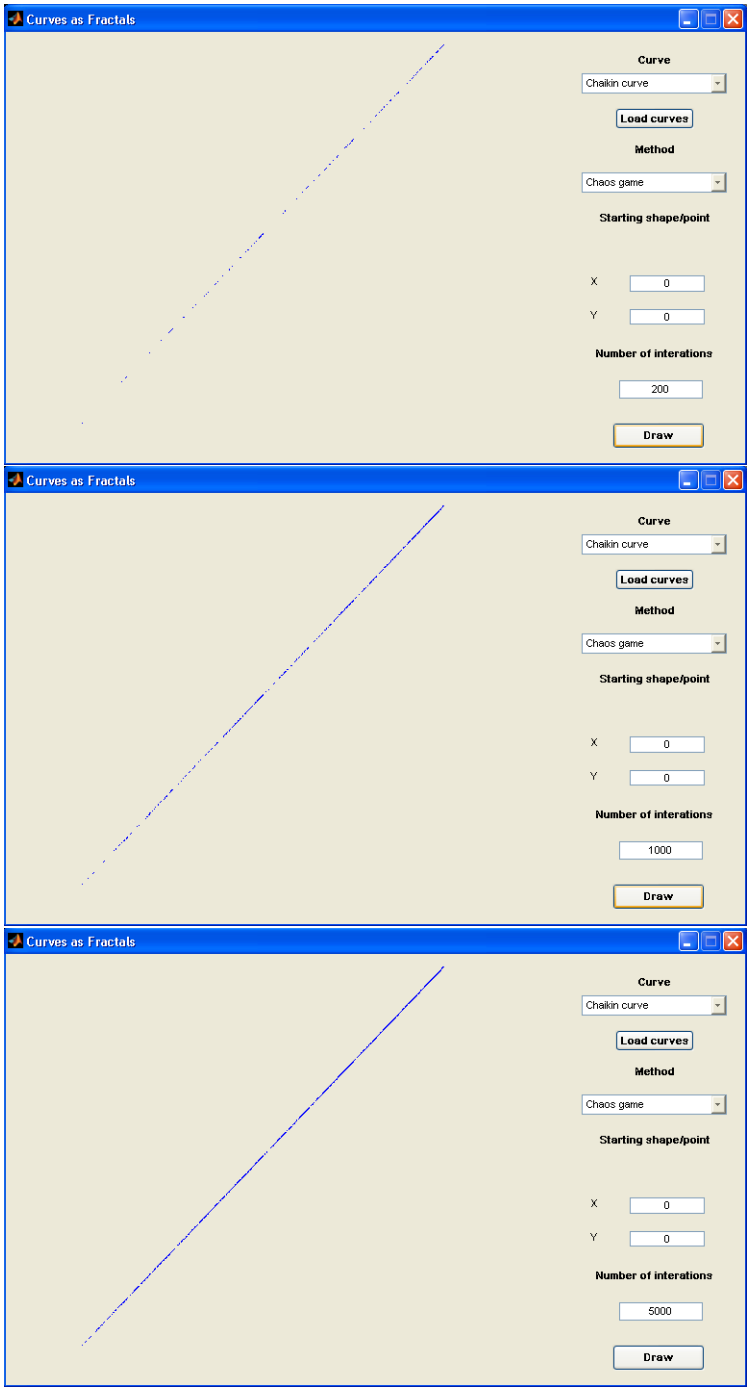

Fig. 13: Single linear segment (Chaikin's case) – Chaos game iterations 200, 1000, 5000

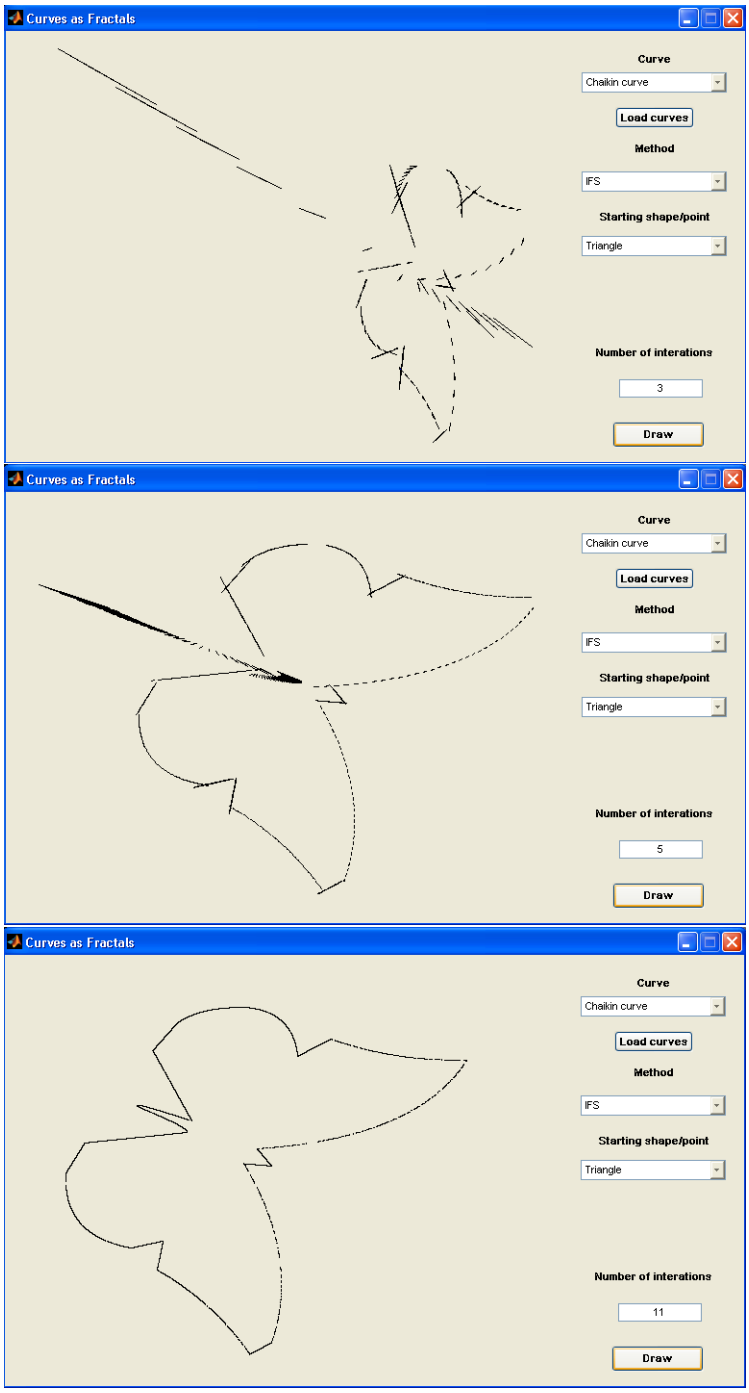

Fig. 14: Butterfly – starts from triangle, iterations 3, 5, 11

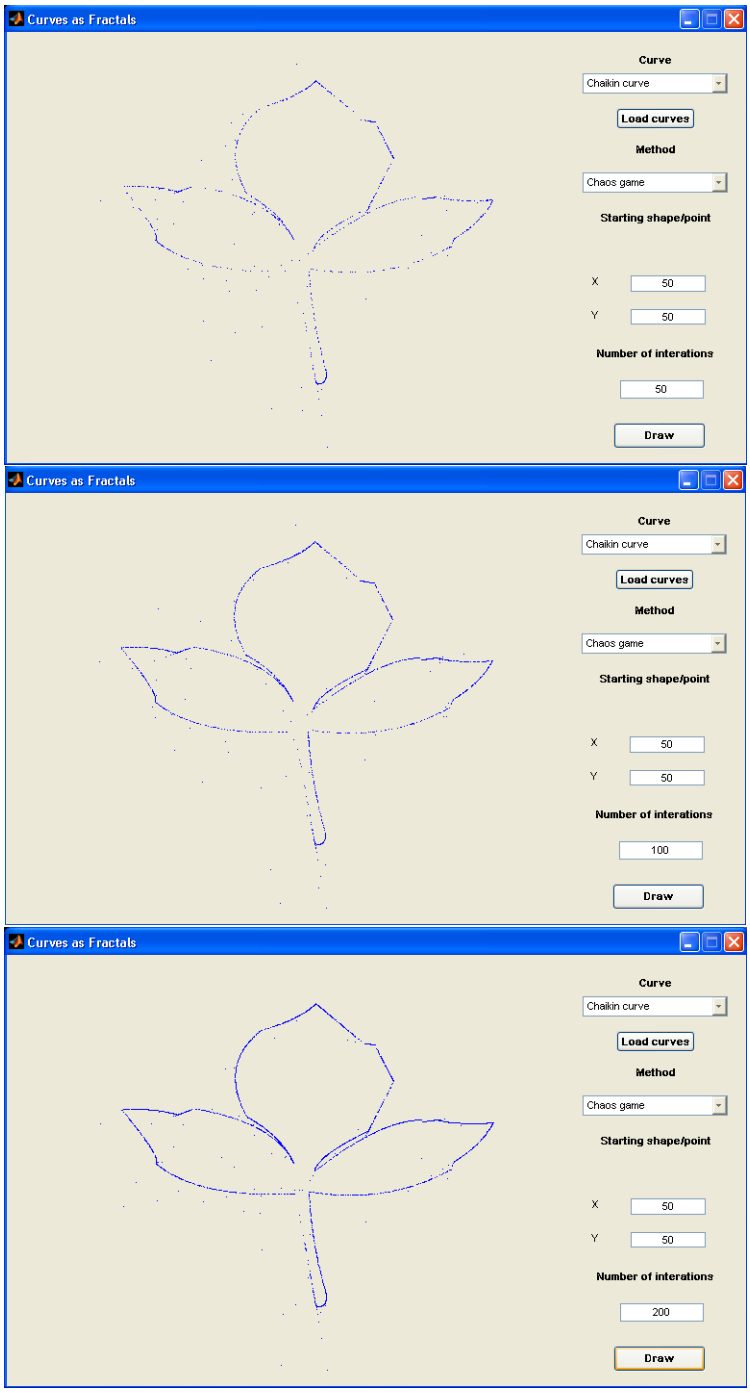

Fig. 15: Box elder – Chaos game iterations 50, 100, 200

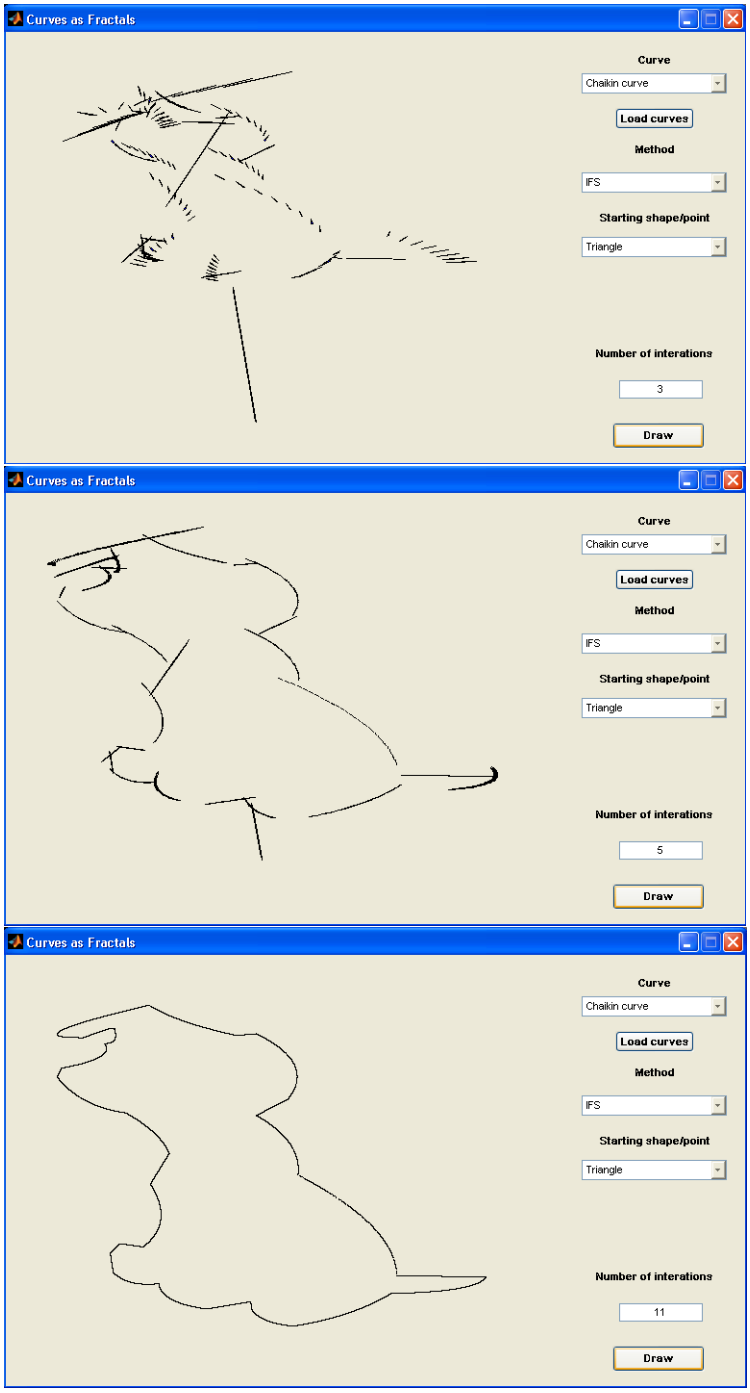

Fig. 16: Dog – starts from triangle, iterations 3, 5, 11