The 3D Chaos Game

Joey Bloom

November 2 2015

Abstract

The Chaos Game is a method of fractal image generation. It is a particle system that draws points one by one according to a set of rules, and the points drawn will approach an attractor which may be a fractal. Many well known fractals in the euclidean plane can be produced by the Chaos Game. I would like to see what images I can produce with the Chaos Game in 3-space or higher dimensions. My goal is to develop an RTICA using HTML5 Canvas and Javascript to display these fractals and allow the user to manipulate the initial conditions to produce their own fractals.

1 The Rules of the Game

1.1 Basic/simple/accessible version

The Chaos Game at a minimum requires a finite set of points $S = \{A_0, A_1, A_2, ..., A_k\}$ and a ratio r where $0 < r < 1$.

1. Choose at random a point $A \in S$.
2. Choose at random a point $B \in S$.
3. Let d represent the distance from A to B. Move point A a distance of rd in the direction of B. That is, $A := rA + (1 - r)B$.
4. Draw the new point A.
5. Repeat from step 2 until a a good enough approximation of the attractor is produced.

1.2 Formal definition

An affine transformation maps each point in a space R to another point in R such that colinearity and distance ratios are preserved. Examples of affine transformations are translations, rotations, dilations, shears, and compositions thereof. Mathematically, an affine transformation T has the form $T(x) = Ax + t$ where A is a matrix and t is a column vector.

An *iterated function system*, or IFS, is a finite set of affine transformations that are all *contractive*. An affine transformation $T(x)$ is contractive if for all displacement vectors a and b, $|T(a)-T(b)| < k|a-b|$ where $0 \leq k \leq 1$. In English, any two points will be strictly closer to each other after T is applied to them. Furthermore, when T is applied iteratively on any two points, the distance between them approaches zero.

1.3 Visualizing IFS

There are two main methods for visualizing an IFS. The first is the *deterministic* method. Given a starting point, each of the transformations in the IFS are applied to the point and the result points are drawn. Then, for each result point from the previous step, each of the transformations in the IFS are applied to the point and those result points are drawn. This tree-like pattern continues to apply all possible sequences of transformations in the IFS of an arbitrary length.

The second method of visualizing an IFS is the chaos game. Where the deterministic method carefully enumerated all possible sequences of transformations, the chaos game randomly chooses one transformation, applies it to the starting point, and then chooses another transformation at random and applies it to that. The chaos game continues to iteratively apply randomly chosen transformations, often producing a desirable image more efficiently than the deterministic approach.

2 RTICA Design

I made a demo of the "basic/simple/accessible" chaos game in two dimensions. It is in my repository at `class198f15/jpbloom2/javascript/chaosGame/simple.html`.

The central feature of my RTICA would be an HTML5 Canvas element on which I would use Javascript to draw fractals. Below the canvas is a drop down menu from which the user can select well-known fractals like the Sierpinski gasket, Sierpinski carpet, and a Barnsley fern. Finally comes the

transformation editor, where the user can edit each of the affine transformations in the IFS.

My idea of the transformation editor is to allow the user to manipulate the affine transformations of the IFS. The user should be able to directly edit the numbers of each affine transformation. I will also include a tool to specify transformations as a composition of translation, rotation, dilation, and shear.

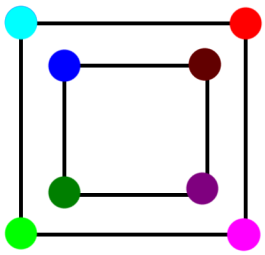

Figure 1: The Transformation Editor

I would visualize each transformation in the transformation editor by showing a unit square/cube and then the same unit square/cube after the transformation has been applied. A sketch is shown in Figure 1 at the right. The square with the light colored vertices is the original and the square with the dark colored vertices is the transformation.

To best show how the Chaos Game builds images point by point, the user should be able to click a "Draw Point" button to cause one more point to be drawn. I will also include a "Run" button which will draw points at a constant rate, specified in a "Points per second" text box available to the user. In the 3D RTICA, I will allow the arrow keys to rotate the fractal image while it is being drawn.

3 Timeline

- F8: Check out "Fractals Everywhere" by Barnsley from Grainger; use this to help formalize my understanding of IFS.
- F9: Proposal 2nd draft completed, with more research into the mathematics of IFS.
- FA: Implement the chaos game in 2D; modify my existing chaosGame2D.html to use affine transformations instead of the basic/simple/accessible version. The user interface for making transformations will come later.
- FB: Extend my implementation of the chaos game to 3D. This will

require using 3D-math.js to project 3D points onto the 2D HTML5 Canvas.

- FC: Visualize the 2D affine transformation editor; make it easy to control with little instruction needed.
- FD: Visualize the 3D affine transformation editor; make it easy to control with little instruction needed.

4 Bibliography

Barnsley, Michael F. "Fractals Everywhere." Academic Press, Inc, 1988. Call Number 516 B267f at Grainger Engineering Library, UIUC.

Barnsley, Michael F. and Andrew Vince. "The Chaos Game on a General Iterated Function System." `http://arxiv.org/pdf/1005.0322v1.pdf`.

Francis, George. "BASIC Pocket Graphics Programs" UIUC Math 198 Hypergraphics 2001 Class Notes. `http://new.math.uiuc.edu/public198/pcPocketry/gasketry/gasket.pdf`.

Hart, John C. "Fractal Modeling." Lecture Slides for CS 418 at UIUC.

Weisstein, Eric W. "Affine Transformation." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/AffineTransformation.html`.