DrawJong 2.0

Michael Hetrick

December 5, 2011

Abstract: *DrawJong 2.0 is a program used to visualize and sonify chaotic attractors. This paper aims to give the reader the history and mathematics behind chaotic attractors before detailing the iterative design process behind DrawJong.*

Committee

Curtis Roads (chair) _____

Clarence Barlow _____

Marcos Novak _____

Matthew Wright _____

1 Introduction and Goals

DrawJong 2.0 is a synthesizer and visualizer based on chaotic mathematics. It was created to serve two distinct needs: the need for an intuitive, yet deep chaotic oscillator, and the need for a real-time renderer of chaotic attractors with an artistic focus..

Up until this point, every readily-available application that I could find rendered chaotic attractors either as an offline process, or as a real-time process with a more mathematical focus and fewer points per frame [1, 2, 3]. When I first started studying chaotic attractors, I found these applications to have hardly any practical value. The reason that these equations hold so much interest stems from the fact that they are highly sensitive to initial conditions. When I changed these conditions on the programs that I could find, I wanted immediate visual feedback that would show me exactly how much I had affected the system.

These programs also tend to only show one chaotic attractor each. There are a wide number of chaotic attractors (a few of which are detailed in the next section), and all of them are distinct and special in their own ways. It seemed rather odd to focus on only one at a time. With this in mind, I began developing a real-time testbed for the exploration of these equations.

I also had the idea that I wanted to focus more on the artistic utility of these attractors. Instead of implementing mathematical overlays (like the aforementioned Wolfram applets), I wanted to have a program that could render the attractors, provide coloring and blending options, and export images while not hiding the attractors under graphs and numbers.

From the very beginning of my time at MAT, I also had the primary goal of developing an interesting and unique audio synthesizer. When I started researching these chaotic equations in depth, I decided to plot their x- and y-coordinates separately as 1D waveforms. These graphs displayed elements of periodicity and contained sections that were similar to sine waves, sawtooth waves, and other shapes commonly used in sound synthesis. With this realization, I decided to combine my goals and attempt to create a synthesizer based entirely on these equations (For a full explanation on how sound is created in DrawJong, please read section 5).

1.1 Aesthetic Goals

My love affair with chaotic attractors started four years ago, when one of my high school friends sent me a Processing sketch of what he had been studying in his math classes. That sketch was the De Jong attractor (detailed in section 2.2). I was immediately taken by many aspects of the images generated by this program. It created figures that defied simple examination. I couldn't decide where my eyes should fall on these figures. They had what appeared to be some sort of periodicity, yet any attempt to follow their movements only made them seem more impressive. They certainly weren't random. There seemed to be something much more organic and inevitable about them.

As I explored these attractors more, I found interesting characteristics unique to each attractor that I found to be visually appealing. The Clifford Attractor, for instance, has states where it becomes very obvious that the equation involves sine waves. You'll see very clear periodicity in some regions, while others seem to follow their own logic. The Duffing Attractor, meanwhile, has a clear symmetry that creates more primitive and rigid shapes. These characteristics and elements of periodicity set the chaotic equations apart from pseudo-random number generators and noise generators.

I was also very excited by the sonified outputs of these attractors. Their chaotic forms (explained more in the next section) create waveforms that alternate between moments of stability and moments that come close to noise. This unpredictability warrants exploration wihtout the prerequisite of deep knowledge. Many of the sounds that I have obtained by accident would be extremely difficult to recreate through more conventional forms of synthesis.

2 Chaotic Attractors

At the heart of DrawJong are five chaotic attractors: De Jong, Clifford, Duffing, Lorenz, and Rossler. The OS X version of DrawJong contains a sixth attractor. Each of these chaotic attractors is a set of equations that exhibit unpredictable behavior.

2.1 A Brief Introduction to Chaotic Attractors

2.1.1 Attractors

Chaotic attractors are a very specific subset of attractors. An attractor is, in its simplest definition, a set towards which all points of a dynamic system move over time. This attracting set can be any geometric form including, at its simplest, a point. The shape of chaotic attractors is described in section 2.1.3.

For a visual example, consider the following pictures of the De Jong attractor:

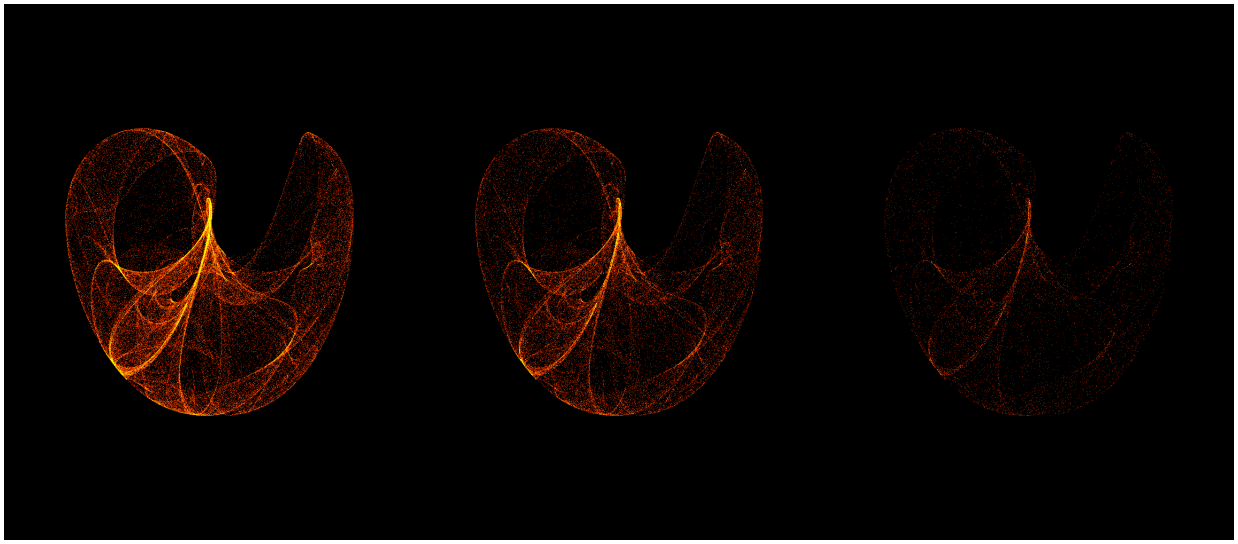


Fig 2.1.1: (Left to Right) A De Jong Attractor's first 100,000 Points; A De Jong Attractor's first 50,000 Points; A De Jong Attractor's first 10,000 Points.

As you can see, the figure looks roughly the same with 10,000, 50,000, or 100,000 points. No matter how many points you add to the system, they will all tend towards the same positions.

A simple attractor is a Fixed Point Attractor. A Fixed Point Attractor is a dynamical system that will end up at a point described by $f(x) = x$ (which is the definition of a fixed point). For an example, consider the following equation:

$$x_n = cos(x_{n-1})$$

In this equation, no matter what choice of real number is used for x_1, the resulting system will always head towards the point .739085133. This point is a fixed point, as cos(.739085133) = .739085133.

Another type of simple attractor is a limit cycle. A limit cycle is simply a periodic orbit (A pendulum is a perfect example of this).

2.1.2 Chaos

The particular attractors used by DrawJong are called "chaotic," as their shapes cannot be predicted until their points are actually calculated, and depend entirely on initial conditions. For a better idea of this, consider a more predictable equation like $f(x) = a * sin(bx)$. If a is affected, the amplitude of the sine wave increases, while b will affect frequency. However, no matter how much a and b are changed, the end result will always be predictable.

Meanwhile consider the following three images of the De Jong attractor:

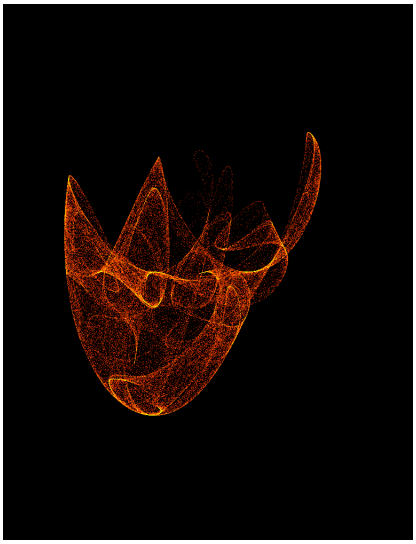


Fig 2.1.2.1: A De Jong Attractor with coefficients a = 1.549, b = 1.104, c = 2.400, d = -2.100

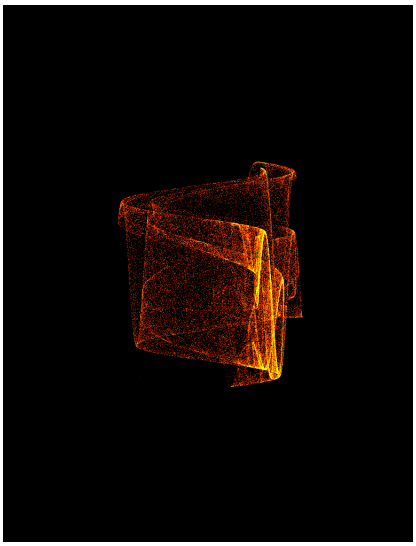


Fig 2.1.2.2: A De Jong Attractor with coefficients a = 0.208, b = -3.447, c = -0.482, d = -3.218

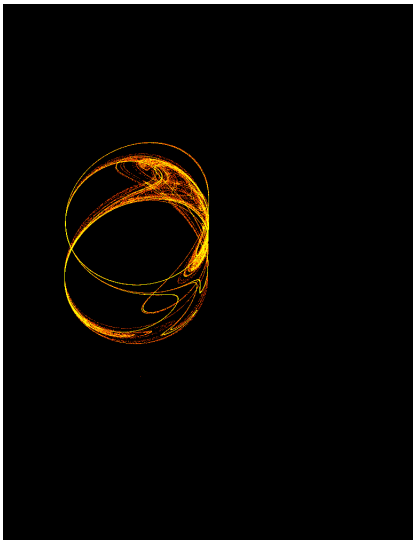


Fig 2.1.2.3: A De Jong Attractor with coefficients a = -3.438, b = 0.088, c = -0.482, d = -3.218

Each of these three images uses the same two equations, yet the changes to their initial conditions cause wildly different figures to appear (If you notice, only the first two coefficients change in the second and third image). It is

important to note that these images are *static* and *deterministic*. The same initial conditions will always result in the same results, and these points do not change with time. However, the fact that these results are deterministic does not change that they can't be predicted before being calculated. This is called "deterministic chaos".

Examples of real-life non-chaotic attractors include a mass spring, in which all points on a spring are pulled to the mass at the end of it. Another example is a metronome, which sticks to a rigid, repetitive limit cycle. In both of these cases, the geometry of the attractor can be represented by a Euclidean shape (The mass spring is a line, the metronome is an arc). The attractors rendered by DrawJong are called "strange" attractors.

2.1.3 Strange Attractors and the Hausdorff Dimension

A strange attractor is defined as an attractor with a non-integer dimension. In Euclidean geometry, every shape has an integer dimension (i.e. a line has dimension 1, a square 2, a cube 3). This integer represents the number of coordinates required to describe a point on this shape. This can be done, as Euclidean shapes are all of finite length. Fractals, however, present an issue, as they are of potentially infinite length. Because of this, they are measured by a value known as the Hausdorff dimension [23]. This can be calculated by the following equation:

$$N = r^D$$

Here, r is the number of times a figure is split in each spatial dimension, N is the number of resulting pieces, and D is the Hausdorff Dimension. For the easiest example, consider a square. If we split it in half in each spatial direction (length and width), we would have four resulting figures. Here, r is 2, as we halved the square in each direction, and N is 4, as we have four pieces. By this, we can calculate that the Hausdorff dimension of the square is 2 (which, as a two-dimensional object, makes perfect sense). All Euclidean figures have Hausdorff dimensions that are integers. Fractals have real numbers. In essence, the reason that these attractors are called "strange" attractors is because all points are being attracted to a figure that is ultimately a fractal.

All of the attractors in DrawJong are examples of strange attractors, and therefore have non-integer Hausdorff dimensions.

2.2 De Jong

The De Jong attractor is a two-dimensional attractor, first described by Peter De Jong [15]. The attractor is described by the following equations:

$$x_n = sin(a * y_{n-1}) - cos(b * x_{n-1})$$

$$y_n = sin(c * x_{n-1}) - cos(d * y_{n-1})$$

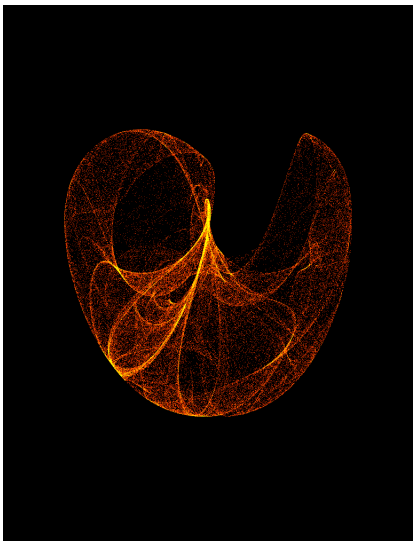


Fig. 2.2.2: A De Jong Attractor with coefficients a = 1.400, b = -2.300, c = 2.400, d = -2.100

It is important to note that for this and for all other attractors used by DrawJong, $x_0 = y_0 = z_0 = 0$.

2.3 Clifford

The Clifford attractor is a two-dimensional attractor, first described by Clifford Pickover (It is also sometimes called a "Pickover Attractor") [14]. The attractor is described by the following equations:

$$x_n = sin(a * y_{n-1}) + c * cos(a * x_{n-1})$$

$$y_n = sin(b * x_{n-1}) + d * cos(b * y_{n-1})$$

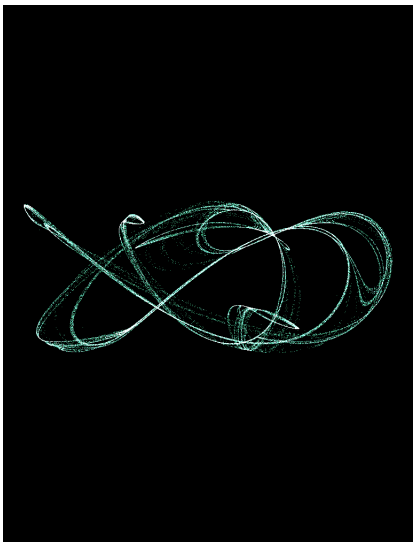


Fig. 2.2.3: A Clifford Attractor with coefficients a = 1.094, b = 1.689, c = 2.266, d = -0.391

Dr. Pickover introduced this attractor in his book *Chaos in Wonderland*. He presents it as a simplified way for understanding chaotic systems.

2.4 Duffing

The Duffing attractor is a two-dimensional attractor, based on equations first described by Georg Duffing [7]. The attractor is described by the following equations:

$$x_n = y_{n-1}$$

$$y_n = x_{n-1} - x_{n-1}^3 - a * y_{n-1} + b * cos(c * n)$$

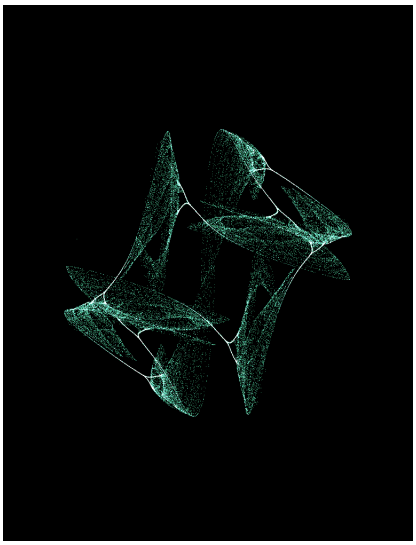


Fig. 2.2.4: A Duffing Attractor with coefficients a = .351, b = -1.037, c = 0.788

The Duffing attractor is the only attractor in DrawJong that occasionally renders coordinates that go towards infinity. Whenever the attractor provides a number that is interpreted as infinity or NaN, the screen will go blank.

2.5 Lorenz

The Lorenz attractor is a three-dimensional attractor that was first described by Edward Lorenz [10]. The attractor is described by the following equations:

$$x_n = x_{n-1} + a * (b * (y_{n-1} - x_{n-1}))$$

$$y_n = y_{n-1} + a * (x_{n-1} * (c - z_{n-1}) - y_{n-1})$$

$$z_n = z_{n-1} + a * (x_{n-1} * y_{n-1} - d * z_{n-1})$$

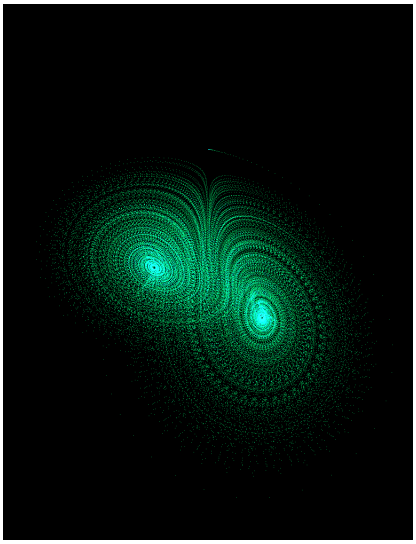


Fig. 2.2.5: A Lorenz Attractor with coefficients $a = 0.010$, $b = 10$, $c = 28$, $d = 2.667$

The Lorenz attractor, unlike the other attractors in DrawJong, was modeled after atmospheric phenomena. This attractor was later proven to show up in nature [22]. The Lorenz's shape stays relatively consistent, and is identified by its two prominent manifolds, also known as "butterfly wings".

2.6 Rossler

The Rossler attractor is a three-dimensional attractor that was first described by Otto Rossler [19]. The attractor is described by the following equations:

$$x_n = -y_{n-1} - z_{n-1}$$

$$y_n = x_{n-1} + a * y_{n-1}$$

$$z_n = b + z_{n-1} * (x_{n-1} - c)$$

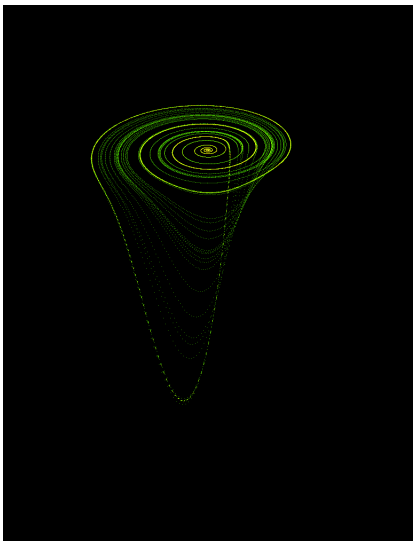


Fig. 2.2.6: A Rossler Attractor with coefficients $a = 0.015$, $b = 0.200$, $c = 0.200$, $d = 5.700$

Like the Lorenz, the Rossler maintains a relatively distinct shape. However, the Rossler attractor usually contains only one manifold. This manifold is known as the "Mobius band".

2.7 Pickover 3D

DrawJong OS X introduces a three-dimensional attractor that I'm calling Pickover 3D. This attractor is based off of a formula relatively hidden in an appendix of Clifford Pickover's *Chaos in Wonderland*. Its five coefficients make it the most complex attractor currently in DrawJong. It is described by the following equations:

$$x_n = sin(ay_{n-1}) - z_{n-1} * cos(bx_{n-1})$$

$$y_n = z_{n-1} * sin(cx_{n-1}) - cos(dy_{n-1})$$

$$z_n = e * sin(x_{n-1})$$

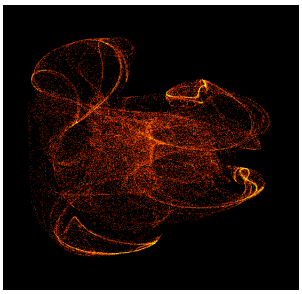


Fig. 2.2.7: A Pickover 3D Attractor with coefficients a = 2.24, b = 0.43, c = -0.65, d = -2.10, e = 1.00

2.8 Henon Map

The Henon Map is a function that was briefly implemented in DrawJong. It was first described by Michel Henon using the following equations [9]:

$$x_n = y_{n-1} + 1 - a * x_{n-1}^2$$

$$y_n = b * x_{n-1}$$

The Henon Map was removed for a number of reasons. Among them are the fact that it did not provide interesting audio results. It also contained a very limited range of parameters where any image would appear. The system often provides coordinates that quickly head towards infinity (even more so than the Duffing Attractor). When the system is relatively stable, the visual results tended to all look very similar.

Fig. 2.2.8: A Henon Map with coefficients $a = 1.4$, $b = 0.3$

2.9 Prior Work

Chaotic equations are not unusual in music both acoustic and electronic. Many researchers and composers have pursued methods of sonifying chaotic equations or using them to create musical sequences. J.C. Sprott wrote the following:

> *Chaotic maps can also be used to produce a crude kind of computer music. For a two-dimensional map, X might be used to control the pitch and y the duration of each note. The result is a not-displeasing though alien-sounding form of music that might appeal to those with exotic musical tastes [20].*

In this example, Sprott has used the two separate axes to create chaotic sequences for music. Researcher/artist Paul Bourke created a similar sonification of the Duffing Oscillator, mapping x and y values to separate MIDI sequences [5].

The electronic work of Iannis Xenakis, especially his GENDY instrument, should certainly be considered as a reference point. GENDY is a software system that use stochastic processes to generate unpredictable waveforms [11]. However, it is important to note that GENDY is based on stochastic (i.e. non-deterministic) processes, while DrawJong is built upon chaotic (deterministic) processes.

Another example of a chaotic system being used to generate control signals can be found in the work of Eduardo Reck Miranda, who used cellular automata to control the parameters of a granular synthesizer [12]. Such techniques have become increasingly popular. In 2005, Native Instruments included in the fifth version of their flagship Reaktor environment a drum machine sequenced by Conway's Game of Life [13].

Oscillators based on chaotic mathematics have been explored since the early 1990s [17]. Researcher Mark Havryliv recently published a paper on a system that he designed for improvising sounds with chaotic oscillators [8]. His method requires audio input that is influenced by chaotic equations, instead of the equations themselves being the sole source of all audio.

3 User Interface

3.1 Main Screen

When a user opens DrawJong, they will see a chaotic attractor, two waveforms, and six buttons: Coeffs, Visual, Synth, Screenshot, Help, and Credits. The top waveform shows the x-coordinate values of the first 2,048 points of the current attractor, while the bottom waveform shows the y-coordinate values. The majority of the screen is dedicated to the attractor.

This menu design has been one of the only things to remain constant since DrawJong 1.0. It was designed to not interfere with the visual image, and to teach the user the underlying complexity piece-by-piece.

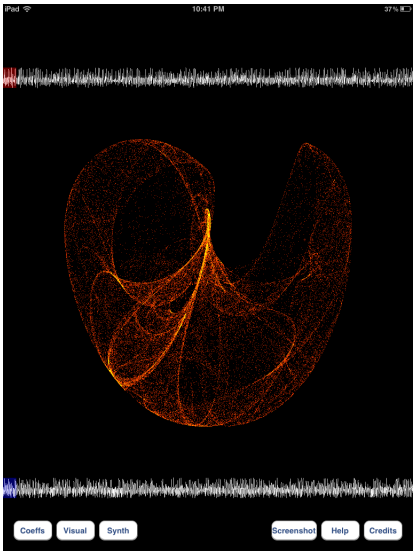


Fig. 3.1.1: The main DrawJong iOS interface

3.2 Buttons and Menus

When a user touches any of the first three buttons, a pop-up menu appears. The buttons and menus are laid out in the order that the user should use them if they are new to chaotic attractors. The very first menu is "Coeffs," which contains the initial coefficients of the current attractor. These controls appear first, as they have the most drastic effect. Changing these controls will affect both visuals and sound. From this menu, a user can also change the position of the attractor on the screen.

The next menu, "Visual," contains color controls, alpha blending, and the equation selector. Aside from the equation selector, these controls have less of an impact than the controls on the Coefficients menu. The equation selector really could have gone either here or under Coefficients, but I chose to place the it on this menu as I felt that it shouldn't be the very first thing a new user sees.

The final menu, "Sound," holds controls for coarse and fine frequency, FM mode, FM amount, gain, and wavetable sizes. I felt that this menu should go last, as many people who use DrawJong will not be familiar with these terms.

There are three more buttons. "Screenshot" saves the current image (minus the buttons) to the user's photo album. "Help" brings up a comprehensive PDF file that lays out the mathematics behind DrawJong, along with detailed help for every control and how to set up OSC. Finally, "Credits" brings up a pop-up menu with all relevant thank-yous.

3.3 Gesture Control

When DrawJong was presented in a public setting for the first time (at MAT's 2011 End of Year Show), its greatest weakness became immediately apparent. Whenever a new user was greeted with the interface, the first thing they did was attempt to manipulate the figure by dragging their fingers across the screen. The menu controls had been a holdover from the very first OS X prototype of DrawJong, and the interface had never fully been updated to accommodate the paradigm shift.

Version 1.2 was programmed and released shortly after this event. With this release, DrawJong received full gestural control over many variables. One finger drags will directly manipulate the first two coefficients of the current attractor, while two finger drags will edit the last two coefficients. These drags are received based on finger coordinates, where the top/left of the screen is the minimum value, and the bottom/right of the screen is the maximum. This is instead of working in an additive nature (i.e. dragging it a certain distance will add a certain value).

This provides two distinct advantages: First, a user can find zones of interest, and can remember regions where their favorite visual and sound combinations occur. Second, it permits a level of discontinuity. Instead of legato morphing, a user can completely change the sound just by sliding their finger to a new region. Simply tapping once on a new region will not change the coefficients. This was something that was tested, but was ultimately determined to be extremely

inconvenient, as errant finger taps occur frequently.

A user can also "pinch" the screen to zoom in and out of the attractor. This is a very common gesture that occurs in many iOS apps. Finally, tapping will randomize the color selections. A single-finger double-tap will randomize the background color, while a two-finger double-tap will randomize the attractor color.

4 Visuals

4.1 Rendering

DrawJong renders up to the first 100,000 points of an attractor per frame. The user can turn down the number of points based on the speed of the device currently running DrawJong.

Point mode is very straightforward. Each point of the attractor is represented by a pixel. To create more interesting images, alpha blending is turned on by default. Alpha blending effectively decreases the brightness of each pixel, but also makes them slightly transparent. This emphasizes denser areas of the attractors, and gives the two-dimensional attractors the illusion of depth. The intensity of the alpha blending can be adjusted or turned off entirely by the user. The user also has control over the color of both the points and the background (given as RGB values).

4.1.1 Line Mode

DrawJong contains an alternate "Line Mode" that connects points in sets of two in their render order. This mode works for all attractors included in DrawJong, and does not affect sound generation in any way. Like point mode, alpha blending can be enabled. Unlike point mode, there can only be up to 5,000 lines on the screen per frame.

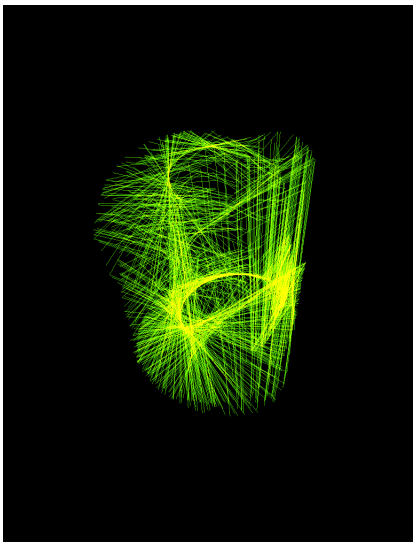


Fig. 4.1.1.1: A De Jong Attractor in Line Mode

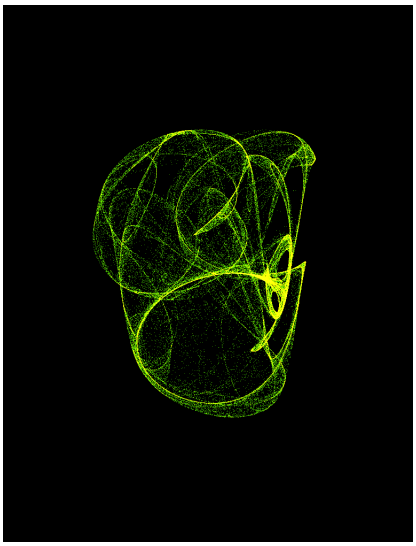


Fig. 4.1.1.2: The same De Jong Attractor in Point Mode

4.1.2 Animated Mode

The "Animated Mode" came out of a programming mistake when creating the first version of DrawJong. The results proved to be interesting, and were implemented as a separate mode in the first official patch for DrawJong.

The "animated" equation for the De Jong attractor is as follows:

$$x_n = sin(a * y_n) - cos(b * x_n)$$

$$y_n = sin(c * x_n) - cos(d * y_n)$$

The "animated" equation for the Clifford attractor is as follows:

$$x_n = sin(a * y_n) + c * cos(a * x_n)$$

$$y_n = sin(b * x_n) + d * cos(b * y_n)$$

As you can see, these equations are not reliant on previous values, but rather current values. Because of this, the animated systems are quite unstable, often

exhibiting states of decay. With the right coefficients, however, they can be used to attain images and sounds that change over a period of time.

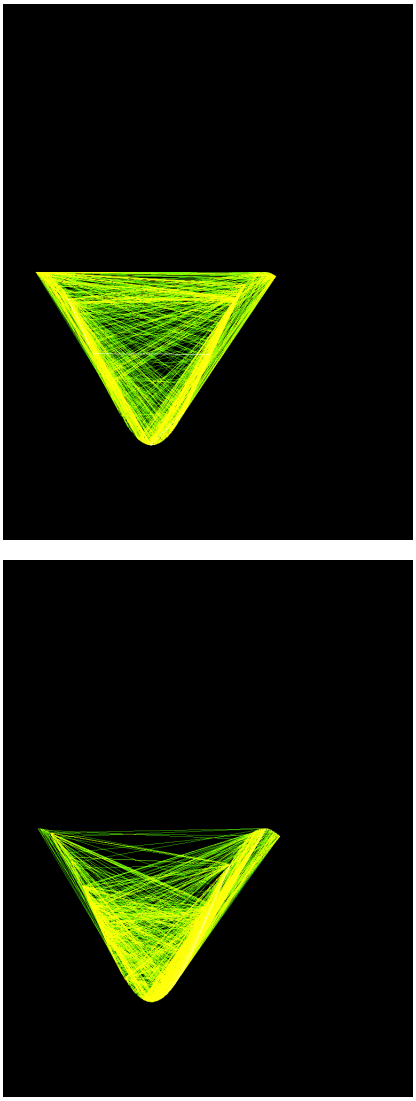


Fig. 3.1.2.1: Two frames from a De Jong Attractor rendered in both Line Mode and Animated Mode. These two frames show a rapid decay of the figure.

4.2 Saving Images

Since the very first iOS release, DrawJong has had the ability to save images of the currently rendered attractor (minus the interface overlay). However,

the image-saving method used a built-in iOS method that saved the images as compressed .jpg files. This created annoying artifacts on the saved images, and it made the feature appear superfluous.

When iOS 5 came out, image-saving became even more broken. Images saved on iOS 5 using DrawJong 1.3 ended up with either an all-white or all-black background. The color of the attractor was affected by the background color, even if it didn't show up (for example, a red attractor on a yellow background was saved as a green attractor on a white background).

DrawJong 2.0 required a large rewrite of the image-saving code to accommodate the iOS 5 changes. In this version, it saves the background without any alpha information to a temporary file. It then writes the attractor and waveform graphic information (with full alpha information) on top of this. After the image is created, it forces a save as a .png file to prevent any artifacting.

5 Sound

5.1 Wavetable Synthesis

In DrawJong, all five attractors are sonified through wavetable synthesis [18]. To achieve this, there are two wavetables: one for the x-coordinates of every point rendered in the current attractor, and another wavetable for all of the y-coordinates.

Let's look at the first two points of the De Jong attractor as an example:

$$x_0 = 0$$

$$y_0 = 0$$

$$x_1 = sin(a * 0) - cos(b * 0) = sin(0) - cos(0) = 0 - 1 = -1$$

$$y_1 = sin(c * 0) - cos(d * 0) = sin(0) - cos(0) = 0 - 1 = -1$$

In this case, the values of the first two samples written to both wavetables are 0, then -1. For the next point, the values of both coordinates will start to differ greatly, based on the user's selected coefficients.

Because of the great variation of possible wavetables (i.e. some attractors exhibit much greater change at quicker rates than others), the user has control over the size of the current wavetable, which must be a power of two. Larger wavetables, for instance, are very noisy on the De Jong and Clifford attractors. On "slower" attractors like the Lorenz and Rossler, a large wavetable is almost required, as variations between coordinates happen at a much different rate.

5.2 FM Synthesis

DrawJong also implements FM synthesis. This works in conjunction with the wavetables. The user can select which wavetable is the carrier, which wavetable is the modulator, and the FM intensity. At lower frequencies and with small table sizes, this can be used as a primitive sequencer, as melodies and rhythms will start to appear. At higher frequencies, many different tones and textures can be achieved.

FM synthesis is achieved as follows: The amplitude of the modulator signal (-1/+1) is multiplied by the FM intensity (0-1,000). The resulting amount is then added to the carrier's given frequency to calculate the carrier's new frequency. This process occurs for every sample calculated.

5.3 *gamma*

All sound synthesis is done through *gamma*[1], a library written by Lance Putnam at MAT [16]. In this implementation, wavetable synthesis is achieved by using the Sweep<> oscillator to scan the current wavetable.

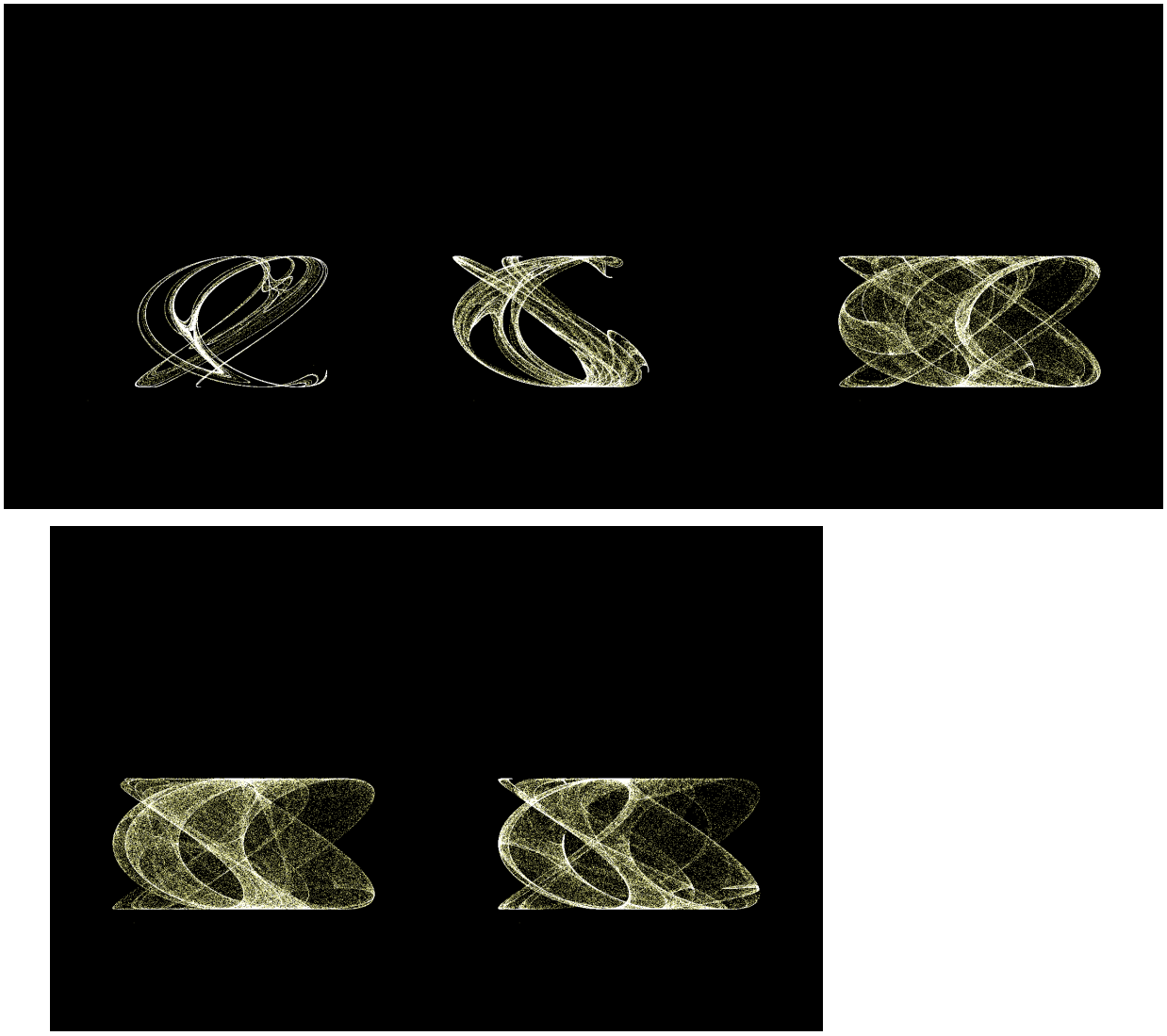
[1] *Note for programmers: gamma* does not include a reference on how to compile it for iOS. When gamma is compiled, it relies on other libraries being present, along with some OS-specific files. To compile gamma for iOS, I removed FFT-related .cpp files (which require the FFTW library), File.cpp and SoundFile.cpp (which rely on the presence of libsndfile), Recorder.cpp and AudioIO.cpp (which rely on PortAudio), and Conversion.cpp and Print.cpp (which I removed to conserve space). The main Objective C file that imports gamma must be renamed to *.mm (where * is the file's name). ".mm" signals to the compiler that an Objective C file will reference a C++ file.

6 Programming Challenges

6.1 Constraining Values

One of the harder design decisions was adding constraints to the coefficient values of each attractor. DrawJong wasn't intended to be a completely scientific tool, but I wanted to make sure that each attractor was fairly represented. In the case of the Rossler and Lorenz attractors, the constraints were chosen to maintain the classic figure of each attractor and avoid values that force the attractor towards infinity. In the case of the Duffing attractor, a balance had to be struck between visual variability and avoiding infinite values. The infinite values still appear, but are infrequent compared to other sets of constraints.

The De Jong and Clifford values were chosen based on the diminishing returns of larger and larger values. As you can see in the following images, increasing coefficients past certain values will only create minor variations of an attractor, instead of providing unpredictable and new images.



(Left to Right)

Fig. 6.1.1: Coefficients a =-0.918, b = -3.340, c = 2.409, d = -0.020

Fig. 6.1.2: Coefficients a = 1.449, b = -3.340, c = 2.409, d = -0.020

Fig. 6.1.3: Coefficients a = 2.695, b = -3.340, c = 2.409, d = -0.020

Fig. 6.1.4: Coefficients a = 3.617, b = -3.340, c = 2.409, d = -0.020

Fig. 6.1.5: Coefficients a = 4.215, b = -3.340, c = 2.409, d = -0.020

6.2 The Scaling Algorithm

One of the greatest hurdles in bringing multiple attractors into DrawJong was designing an efficient scaling algorithm for the wavetables. Scaling wavetables for only the De Jong attractor was easy. As the De Jong equation's coordinates never exceed -2 or 2, all wavetable values were simply 1/2 of the coordinate values..

However, consider the Clifford Attractor. Because the coefficients exist outside of the sine and cosine functions, the coordinates have the potential to be as large as the largest c or d coefficient. In the case of DrawJong restricting the maximum/minimum values of these two coefficients, the Clifford Attractor's bounds are at -6 and 6. The possibility exists just to take a sixth of every wavetable value, but this creates difficulties when considering future expansion. If the maximum/minimum restrictions were to change, the "magic number" for scaling would need to change as well. This doesn't even factor in the other three attractors, which are much harder to predict.

With all of this in mind, the decision was made to quickly scan the wavetable for its greatest value, and scale both the image and the waveform by one divided by that value. To prevent any weird, sudden scaling issues, a large number based on the current attractor is provided to this method (For example, in the case of the Clifford Attractor, the number is given as one plus the maximum coefficient restriction). If a value larger than this preset number is found, it becomes the new preset number until a new attractor is loaded.

Now, this may seem to be mathematically unfaithful to the equations. In terms of the overall usage experience, though, the scaling algorithm is barely noticed. If the scaling preset is actually exceeded, previous settings may seem "zoomed out," but this can be fixed by zooming the OpenGL camera back in on the attractor. No severe volume gain losses have been detected. If it seems much less efficient than the previous method, in reality it isn't, as the previous method occurred at audio rate, while this method occurs at video rate (more on this in the next section). Furthermore, the constant is only generated once for the De Jong (2) and Clifford attractors.

Aside from this minor drawback, this scaling method creates the benefit of maximum expandability for future DrawJong updates. DrawJong can, from now on, receive more unbounded attractors or coefficient bounding changes, and the scaling algorithm will not require another rewrite.

6.3 Audio Rate vs. Video Rate

DrawJong contains two separate callback methods–one for audio, and one for video. The audio callback implements Apple's AudioUnit technology and runs at the device's sampling rate (44.1 kHz). The video callback runs on OpenGLES and runs at however many frames the video card can render per second. This is an important distinction, as the audio callback runs at a higher priority, and never changes speed. Video slowdown does not greatly affect the user experience in the way that an audio slowdown would.

The audio callback contains the fewest instructions possible. gamma's Sweep<> oscillator checks the current wavetables in memory and saves their values as temporary variables. These variables are used for calculating any frequency modulation that will occur (Calling upon the Sweep<> oscillator for this information would force it to jump to the next sample). After these calculations, the current samples are sent to be played through the speaker.

The video callback contains almost every other important real-time update in DrawJong. In this method, the attractor is calculated, checked for extreme values (infinity, NaN, and numbers outside of the expected range of the attractor), and scaled. From this large array of calculations, two wavetable arrays are created. During the video callback, the program also checks for OSC updates and gesture inputs. These input changes are then used for the next rendered frame.

6.4 Vertex Arrays

The very first iteration of DrawJong used an inefficient for-loop rendering method known to OpenGL programmers as "Immediate Mode". In this method, individual instructions are sent one-by-one to the video card. In the case of an attractor being displayed with 50,000 points, this would result in 50,000 separate GL_POINTs being sent to the video card for every single frame. In pursuit of the goal of real-time attractors with many points, I required a new rendering method.

At the suggestion of Angus Forbes, an upgraded Vertex Array-based method was implemented. In this method, instead of sending thousands of individual instructions to the video card, an array to be rendered is passed. Each point that is being rendered is in the array as a set of two numbers: x-coordinate and y-coordinate (and in the case of the 3D-attractors, a z-coordinate).

The benefits of this method are large. In the original DrawJong, approximately 25,000 points could be rendered before frames were skipped. With the vertex arrays in place, about 100,000 points could be rendered smoothly. In addition to this, it saves many lines of code. DrawJong's "Line Mode" renders the attractors as more solid figures by connecting points in order. To achieve this in the point-by-point rendering method, a separate for-loop had to be written that connected points in order. With Vertex Arrays, OpenGLES only needs the flag GL_LINES to achieve the exact same effect.

Because of this, many other rendering primitives were experimented with, including triangles, circles, planes, and more. In the interest of aesthetics, only Line Mode and Point Mode were kept. The other shapes ended up completely obscuring the shape of the attractors, and more or less appeared as formless blobs.

After Vertex Arrays were implemented for the attractor rendering, they ended up in every aspect of visual rendering, including the on-screen wavetables and the wavetable selection boxes.

6.5 Separating 2D and 3D Methods

DrawJong's code underwent a complete rewrite and modularization during the upgrade to 2.0. Before 2.0, DrawJong only supported the De Jong attractor. As such, the methods for calculating the equation were built into the rendering methods. There was a simple Boolean check to decide whether to use De Jong or Clifford equations, but this Boolean did not have user access in DrawJong 1.0, as a proper scaling method for the Clifford Attractor had not been worked out yet.

The De Jong and Clifford attractors were split into two new .c and .h files each. It is important to state that these are not new C++ classes, but just regular C files that contain different methods for each attractor.

After the original two attractors were successfully working within the new structure, all Boolean checks were replaced with a switch, and the Henon Map was added. As described in Section 2.6, this map was removed from DrawJong. The Duffing Attractor then replaced it.

At this point, the Rossler and Lorenz equations were the two most famous chaotic attractors not present in DrawJong. To finally accommodate them, the main rendering method was split into three methods: a two-dimensional renderer, a three dimension renderer, and one method with common instruc-

tions shared by both methods. At this point in time, adding new attractors to DrawJong consists of creating a file with a method containing the attractor's equations, and telling DrawJong whether to use the two-dimensional or three-dimensional method.

6.6 C, C++, and Objective C

DrawJong was written using a blend of C, Objective C, and C++. All memory management, interface loading, and general iOS programming is done in Objective C. The common attractor rendering methods are written in C, while gamma and OSCPack are written in C++.

6.7 OSC Support

OSC was added in the third major patch to DrawJong, and it also represented the single biggest roadblock in the development process. Liblo was initially selected as DrawJong's OSC library, as it is written in C, and can be added to projects very easily. However, Liblo's licensing scheme prevents commercial works from using it unless access is provided to the .a files for end-user editing. This creates a major conflict on iOS devices where users cannot edit internal files, as there's no user-accessible file system to begin with. To complicate matters, many programmers have worked on Liblo, and every single programmer has differing opinions on using Liblo on iOS devices, ranging from *laissez-faire* to full-on hatred of Apple's policies.

After a few days of waiting for solid permission on using Liblo, I made the decision to switch to OSCPack, a C++ library by Ross Bencina.

The final DrawJong OSC implementation considers the many different message standards sent from different pieces of software and hardware. A Lemur, for instance, has a multislider object that sends to one address (i.e. /slider), but appends the number of each slider onto the address, so the first slider sends to /slider/1. Max/MSP, meanwhile, can send an entire list of variables to one address. DrawJong is capable of receiving messages structured in both ways at the same time. For instance, for the four coefficients, messages can be received either in a list format at "/coefficients" or on an individual variable basis to "/coefficients/1," "/coefficients/2," etc. (which are the standard address suffixes on a Lemur). There is also the option to send on "/coefficients/a," "/coefficients/b," etc.

6.8 OS X Port and Translab Installation

In November 2011, at the behest of Marcos Novak, I ported DrawJong to OS X as a native Cocoa application. This was primarily done to turn DrawJong into an installation inside the TransLab environment at MAT.

For the most part, the port was straightforward. OpenGLES needed to be stripped out and replaced with OpenGL, and the AudioUnit portion of the sound output methods was replaced by PortAudio.

The purpose of the installation in the Translab shifts DrawJong towards a more tangible environment. In this environment, a user holds a sensor that tracks seven quantities: x, y, and z positions along with a four-dimensional quaternion. As the user walks through the room and manipulates the sensor, he manipulates a chaotic attractor in space. Rotating the sensor rotates the attractor, while walking through the room moves the attractor in space (For example, if the user moves towards the right side of the screen, the attractor moves in the same direction). Rotating the sensor also affects the coefficients of the attractor, while position affects oscillator frequencies.

The OS X version also adds the Pickover 3D attractor, and the ability to export 3D attractors as point cloud data. This data can then be used in other 3D modeling programs.

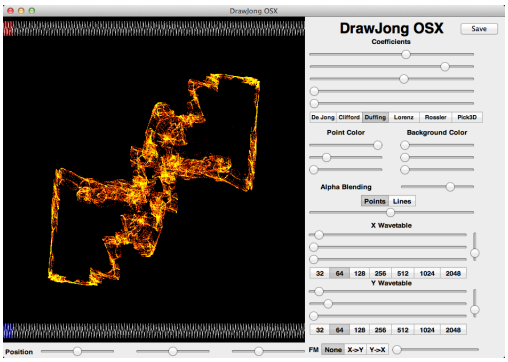


Fig. 6.8.1: The main DrawJong OS X interface

7 Release and Future

7.1 Release and Updates

DrawJong was released on the Apple App Store on May 13th, 2011. It proved to be successful, and peaked at #23 in US iPad music sales. During this period, popular electronic artist Richard Devine tweeted about the app and praised it on an experimental synth forum, saying "I love the interface and the sounds are wicked". After a week of high sales, it settled into a period of consistent daily sales of five to ten apps.

The first major update (1.1) was released two weeks later. This update provided support for iPhone and iPod Touch, and also added a new rendering mode that constantly animated points on screen. The second major update (1.2) added gesture controls. With this, the user no longer had to bring up a menu to change the coefficients of the figure. The third major update (1.3) added full OSC support.

On November 9th, 2011, DrawJong 2.0 was released. In this version, DrawJong received four additional attractors (Clifford, Duffing, Lorenz, and Rossler). In addition to this, many long-standing efficiency bugs were fixed.

7.2 Future Plans

DrawJong's OS X version will ultimately be wrapped as both a VST [21] and an Audio Unit [4]. After this, new attractors are being considered. Further down the road, support for LFOs may be added, so that a user can create an evolving sound without relying on Animation Mode or OSC. CoreMIDI support is also being looked into. Some users have requested support for envelopes, but I feel that this isn't in line with the rest of DrawJong, which encourages constant sound and manipulation.

A large visual update planned for later will contain support for pixel shaders. These will add a lot of pleasant visual effects like multi-color attractors.

Outside of additions to DrawJong, I have been able to show that rendering tens of thousands of chaotic points both visually and sonically is computationally feasible. Knowing this, it seems like generating much smaller wavetables (32-1,024 points) for oscillators would be a great way to expand other applications. A more traditional synthesizer could use these chaotic wavetables for LFOs to create varied modulations, or for audio-rate oscillators to create radical FM tones.

8 Conclusions

In sum, I feel that DrawJong was a success. From an objective standpoint, it meets my original goal of creating a real-time rendering environment of chaotic oscillators. As an audio synthesizer, it meets my goals partway. On one hand, it is capable of creating a wide variety of tones and timbres in a very intuitive manner. On the other, it still lacks many common features like envelopes, LFOs, and a keyboard. While these lacks were part of the design of the program, I know that they can be implemented into a different program with a different focus without compromising the unique sound capabilities. As stated in the previous section, these wavetables could be put to powerful use in an instrument designed for more straightforward Subtractive/FM synthesis.

With all of that being said, it is the only real-time renderer of these equations that is readily available for interested artists and researchers. It is an efficient, well-designed program that can easily take on more equations in future expansions. I've learned more about operating systems, interface designs, and programming languages in the year spent writing this program than in my previous twenty-plus years of using computers. In all, it has helped me to better understand these equations that originally seemed so esoteric while expanding my sample library in the process, and I am happy to see how many other users feel the same.

9 Acknowledgments

First and foremost, I would like to thank my committee: Curtis Roads (Chair), Clarence Barlow, Marcos Novak, and Matthew Wright. They helped me to better articulate much of my research and greatly assisted me in writing this, my first full-length research paper.

DrawJong was inspired by Paul Bourke's comprehensive website [6], and the writings of Clifford Pickover, which detail many varieties of chaotic attractors in a simple-to-understand manner. Lance Putnam deserves credit for his powerful audio library, *gamma* (not to mention his patience with putting up with my questions). Many thanks to Ross Bencina, who not only created an excellent library with OSCPack, but also released it entirely for free. I'd like to thank the teachers who gave me the tools I needed to create DrawJong, including Charlie Roberts, who taught me iOS programming, Angus Forbes, who taught me OpenGL, and Matthew Wright and Ryan McGee, who taught me the dark

art of audio programming.

References

[1] Nodebox: Peter de Jong attractors.

[2] Open Processing: de Jong Attractor. http://www.openprocessing.org/visuals/?visualID=2097.

[3] Wolfram Demonstrations Project: Peter de Jong attractors. http://demonstrations.wolfram.com/PeterDeJongAttractors/.

[4] Apple. The Audio Unit. http://developer.apple.com/library/mac/.

[5] Paul Bourke. Duffing attractor. http://paulbourke.net/fractals/duffing/.

[6] Paul Bourke. Homepage of Paul Bourke. http://paulbourke.net/fractals/.

[7] Georg Duffing. Erzwungene schwingungen bei veränderlicher eigenfrequenz. *F. Vieweg u. Sohn*, 1918.

[8] Mark Havryliv. Composing with chaotic oscillators. *NIME*, 2010.

[9] Michel Henon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50:69–77, 1976.

[10] Edward Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, March 1963.

[11] Sergio Luque. The Stochastic Synthesis of Iannis Xenakis. *Leonardo Music Journal*, 19:77–84, 2009.

[12] Eduardo Reck Miranda. Granular synthesis of sounds by means of a cellular automaton. *Leonardo Music Journal*, 28(4):297–300, 1995.

[13] Native Instruments. *The Future of Sound: 15 Years of Native Instruments*. 2011.

[14] Clifford Pickover. *Chaos in Wonderland: Visual Adventures in a Fractal World*. St. Martin's Press, 1994.

[15] Clifford Pickover. *The Pattern Book: Fractals, Art, and Nature*. World Scientific Publishing Company, Hackensack, NJ, 1995.

[16] Lance Putnam. gamma. http://mat.ucsb.edu/gamma/, 2011.

[17] Curtis Roads. Personal communication.

[18] Curtis Roads. *The Computer Music Tutorial*. MIT Press, 1996.

[19] Otto Rossler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[20] J.C. Sprott. Automatic generation of strange attractors. *Chaos and Fractals: A Computer Graphical Journey*, pages 53–60, 1998.

[21] Steinberg. Our Technologies. http://www.steinberg.net/en/company/technologies.html.

[22] Ian Stewart. The lorenz attractor exists. *Nature*, 2000.

[23] Vanderbilt. Fractals and the Fractal Dimension. http://www.vanderbilt.edu/AnS/psychology/cogsci/chaos/workshop/Fractals.html.