

Aesthetic Emergence in Synesthetic Sculptures

By

Adrian Barret Johnson

B.S. C.S., University of Colorado, 2004

A thesis

submitted to the Graduate Faculty of the University of Colorado at Colorado Springs

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

2007

© Copyright by Adrian Barret Johnson 2007.

Permission to make digital or hard copies of all or parts of this work for personal or classroom use is granted without fee provided that copies are not distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

This thesis for the Master of Science degree by

Adrian Barret Johnson

has been approved for the

Department of Computer Science

By

Sudhanshu K. Semwal, Chair

Edward Chow

Valerie Brodar

Date

Johnson, Adrian Barret (B.S., Computer Science)

Aesthetic Emergence in Synesthetic Sculptures

Thesis directed by Professor Sudhanshu K. Semwal

This thesis describes a means of co-evolving parameters for procedural animation generation. Several experiments were completed in order to find an aesthetically pleasing animation mechanism which could be both modulated by music and have its parameters evolved via genetic algorithms. The selected music visualization is a set of vector fields for the purpose of moving particles whose paths create visual interest. To cause emergence of pleasing patterns in particle behavior, several metrics were incorporated into the fitness function. To reflect the spatial characteristics of the vector fields a multi-dimensional crossover operator was developed.

The most critical aspect in creating a vector field which animates particle paths over time is that the vector field not allow the particles to become static or too tightly clustered, as either indicates the particles will no longer be circulating along pleasing patterns. We are interested in encouraging certain behaviors of the particles without doing so explicitly, and an evolutionary approach will allow for many aesthetically pleasing solutions to be reached. We want to evoke patterns in the particles which suggest motion, suggest continuous geometric surfaces, and create pleasing textures through the spacing between particles. The evolutionary approach will continue to work as additional complexities come into play, such as more modulation sources for the particle motion. If pleasing patterns can be made implicitly with the vector fields in this specific audio-visual composition, it stands to reason that parameters for other

visualizations may be evolved as well and this initial work serves as an argument for computer assisted aesthetic refinement.

Acknowledgements

I would like to thank the following people:

My father, Dana Johnson, for introducing me to electronic music and computer graphics.

My thesis advisor, Dr. Sudhanshu Semwal, for guiding my work.

Dr. Chow and Valerie Brodar for their thorough and detailed feedback.

My friends and family for their unending patience and support.

List of Figures

1. ADSR Envelope
2. A particle system used to simulate fire
3. Rule 30
4. Some patterns found in nature are formed by cellular automata
5. Structure imposed on the swarm through initial position and vector field edges
6. DOF rendering and rhythmic swarm contractions
7. Segmentation of the subject and several variants on erosion and dilation
8. Height map interpretation of video frames
9. Addition of graphic elements to the height map
10. Environment mapping
11. DOF effects with geometry overlay
12. Comparison of depth cues from environment mapping and DOF rendering
13. Early stage of synesthetic sculpture construction
14. Growth of synesthetic sculpture, 10 frame intervals
15. Path crossing causing extrusion
16. Concentric paths
17. Three swarms corresponding to drum components
18. Substantial improvement in fitness between generation 1 and generation 100
19. Generation 1
20. Generation 50
21. Generation 100
22. Generation 1 – limited elitism

- 23. Generation 50 – limited elitism
- 24. Generation 99 – limited elitism
- 25. Generation 100 – limited elitism
- 26. Scenario A – max 7.7
- 27. Scenario B – max 14.9
- 28. Scenario C – max 7.14
- 29. Scenario D – max 4.6

Contents

Abstract	i
Acknowledgements	iii
List of Figures	iv
1. Introduction	1
2. Background	5
3. Visualizations	21
4. Genetic Algorithm Design and Implementation	35
5. Implementation Details	39
6. Results and Observations	42
7. Suggestions for Further Research	56
Bibliography	58
A. Generational Snapshots	60

Chapter 1

Introduction

The link between music consumption and the emotional state of a person is not entirely understood, but a connection between the two has been proven many times [1]. The quality of a live music performance or music recording is related to the artists' ability to connect with the listeners on an emotional level. During some performances large projection displays are used. The motivation for having visual content during a live music performance is to make the experience more immersive and ultimately more emotive. The visuals should be pleasing to the eye taken separately, but also synesthetic when coupled with the music. To address the latter, it is possible to cause the visuals to be modulated by the music. This may be done through MIDI events for the programmed digital instruments, or a spectral analysis of either live or programmed instruments. While the mapping of musical events to envelope parameters may be a challenge, the greater challenge would be defining the aesthetics of the underlying routines. It is the independent aesthetic value of the visuals we seek to address with evolutionary techniques presented in this thesis.

The purpose of this thesis is to explore computer assisted generation of audio-visual art. We are using a section of an original music composition to modulate the motion of particles through vector fields, and desire the vector fields to evolve in order for the particle paths to meet aesthetic criteria. Several other procedural graphics techniques were explored before delving deeper into the use of vector fields. The

determining factors when selecting the visual technique to evolve was a matter of to what degree could genetic algorithms alter the overall aesthetic, the scalability of the technique, and the novelty of the result. The visuals being created are meant to accompany a live music performance and also be included with a DVD release of an album, so the aesthetic had to accommodate the intended use of the visuals. We desired for the visuals to suggest motion, alluding to the melodic motion of the music. We also wanted the visuals to have geometric patterns and shapes. Through the forms and poses woven by the particles, we wanted the particle placement to seem structured rather than random. The visual interest we strove to create within the virtual structures was based on this structure, a delicate sense of “order from entropy.” An aesthetic favoring structure emerged.

We have had an interest in genetic algorithms for quite a while. Though some applications may seem somewhat straight forward - for instance, optimization of flight paths in an unmanned aerial vehicle (UAV) surveillance network - it is much more interesting from a computational standpoint when the fitness of a system is difficult to quantify. For example, how does one define the fitness of the stylistic attributes of an image or a video sequence? What can be called interesting to an observer? Some observations have been made regarding a variety of stroke directions and weights contributing to the immersiveness of a painting [2], and that was taken as inspiration. In parallel to respecting traditional art forms, we must acknowledge emerging media and new types of artists, such as video jockeys (VJs) who synthesize and mix video to accompany and reflect music. VJ culture is becoming increasingly prevalent, with night clubs using multiple plasma, LCD, or projection screens. By having a portfolio of visual

content we hope to be able to perform in venues where multiple senses may be catered to. In order to be able to perform independent of venue equipment we acquired two 42" 1080p LCD screens, and the visuals are being tested on these screens from multiple viewing distances. We would like the content to be visually interesting from five feet away or fifty feet away. Just as impressionist paintings are pleasing from a great distance when the artist's dithering and the interpolation done by the human visual system have an interplay, those same paintings have fascinating brush strokes when viewed up close. Could such an effect be created with procedurally created moving graphics?

Under the Rustcycle moniker, we have explored performance with and without the aid of visual content. When using moving head discharge lamps, lasers, fog and haze machines, strobe lights and washes, it was said several times that the performance was "too big for the venue" indicating the use of lights seemed very professional and immersive. An aspect to consider with promotion and branding of a performance is the degree uniqueness and professionalism – and in many ways developing unique visual content is an extension of creating unique musical content. To be competitive in the music industry on a national scale we felt it was necessary to have a visual display that was more cutting edge than off-the-shelf lighting in order to play into the image of the Rustcycle project. In addition to forming a stronger connection with the audience, we desired a complete audio-visual experience that would warrant some high-profile demonstrations, interviews, and collaborations. This work fits into a strategy for making the performance more unique stylistically, but it is also to allow us to create a more unique type of product. Brian Transeau (who performs under the moniker BT) released "This Binary Universe" as a 5.1 surround album featuring ambient animation. More than

music videos, the visuals are a beautiful backdrop to be enjoyed while focusing on the music, or during a live performance, focus on the musicians. A collection of video artists and animators created the visual content on a per-track basis, lending each track a different feel. The visual content varies from surreal and futuristic imagery to geometric animations to excerpts of home video showing BT's daughter.

The way Trent Reznor gained notoriety by composing his albums solo and hiring out live musicians we would like to compose the music and visuals for an album as a solo endeavor to have complete control of the unity of the vision. Generally artists, author included, embrace collaboration and believe the sum is greater than the sum of its parts, but at times a composition must be created with a singular voice. Due to the ambitious scale of creating the music and visuals for a DVD album, procedural generation of content is a practical avenue to pursue. We are interested in whether evolutionary techniques may be like having an assistant to carry out designs architected by the artist, and whether this is the first step towards having collaborative works created between human intuition and synthetic variation; Kurzweil describes software which at first serves as a sophisticated tool for the artist, but eventually is able to create pleasing works through an understanding of aesthetics [3]. This remains a challenge, and this thesis presented endeavors to move in that direction.

Chapter 2

Background

2.1 Introduction

In this section we will provide background on some of the mechanisms and concepts used to implement the visualizations and evolutionary techniques required by this project. We will examine image processing, attack-decay-sustain-release (ADSR) envelopes, computational aesthetics, synesthesia, particle systems and vector fields, swarming behavior, genetic algorithms (GAs), and emergent behavior. This is not intended as an exhaustive survey, but rather to illustrate the individual items to facilitate understanding of their contributions to the overall whole.

2.2 Image Processing

The image processing we will be dealing with includes some image subtraction and convolution kernels. In image subtraction pixel values at the same X,Y locations in two images are obtained and the difference found. This is carried out for each pixel in the images. Image subtraction can be used for masking out the background to obtain only the pixels the subject is occupying. By treating video as a sequence of images we can take a video frame without the subject and subtract it from a frame containing the subject. If the pixel value difference (calculated by summing the differences of the red, green, and blue components) exceeds a threshold, then the pixel is classified as belonging to the subject

and is kept. The advantage of image subtraction over chroma key masking (where a blue screen or green screen is used) is that an arbitrary video location may be chosen, rather than one where a chroma screen may be used. This also allows for much larger video capture areas. A large outdoor scene would be impractical to mask with a chroma screen. We may also want both the background and subject and just require segmentation of the two. A chroma screen would block out the background unless another video sequence was created after removing the screen. Some disadvantages of image subtraction for subject segmentation are sensitivity to lighting variations and lack of robustness in the case of similarity between subject and background pixel values. In the case of the former, if the subject casts a shadow in the background, the shadowed background region may also be segmented. In the case of the latter, areas where the subject is similar to the background will not be segmented. For instance, on a person with dark hair who is crossing in front of a dark region of the background, the hair will not be segmented. Careful selection of the background location and the appearance of the subject can help to minimize some types of segmentation problems.

Due to the noise inherent from frame to frame in a video sequence further processing of a segmented subject may be necessary. A convolution kernel may be used to filter either the pre-processed image or the pixel mask resulting from image subtraction. A convolution essentially assigns a pixel a new value based on its current value and those of its neighboring pixels. Kernels in a square shape are quite popular, such as a 3x3 arrangement. A simple blurring kernel would assign a pixel value using a summation of $1/9$ the value of the original pixel and its eight contiguous neighbors. This averages the pixel values. To further blur the image, the convolution could be executed

multiple times. Other kernels can be used to sharpen an image or accentuate certain features. Erosion and dilation kernels are very useful for modifying the mask created by image subtraction. An erosion operator will move across the mask and search out pixels with 0 or 1 neighbor pixels and invert the mask value. This would find single masked pixels astray from the subject and eliminate them. A dilation operator will increase the number of pixels segmented. It would find unmasked pixels with 2 or 3 masked neighbors and set the pixel to masked. A combination of dilation and erosion can be used to make the edges of the segmented subject clearer. These operators and others may also be used to distort and stylize the image.

2.3 ADSR Envelopes

From the realm of granular synthesis Attack-Decay-Sustain-Response (ADSR) envelopes are popular for defining how a musical note behaves over time. Such an envelope can be applied to the note volume or other characteristics (Figure 1). Using the example of an ADSR envelope for modulating volume, *attack* is the amount of time the note takes to reach maximal volume once the event is triggered, *decay* is the amount of time the note takes to lessen in volume until the sustain volume is reached, *sustain* is the volume level which is held constant for the remainder of the note, and *release* is the amount of time it takes for the volume to fall from the sustain level to silence. Such envelopes may be applied to visual content as well. In a video synthesizer one could modulate luminosity or blurring rather than volume. In a particle system one could similarly modulate the magnitude turbulence. ADSR envelopes are applicable to many generative processes.

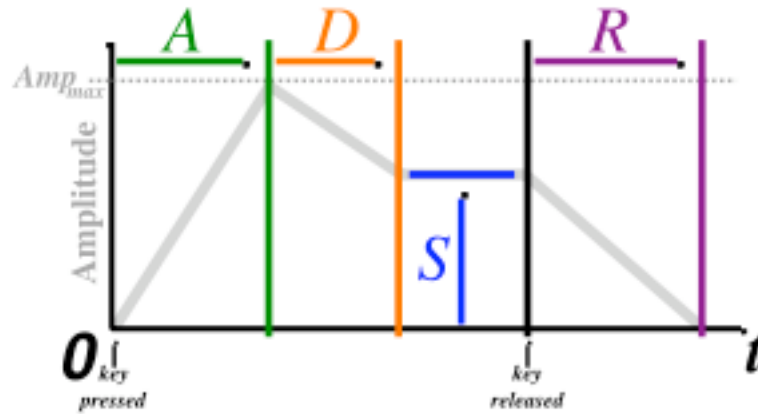


Figure 1. ADSR Envelope

2.4 Computational Aesthetics

Computational aesthetics is a relatively new field, incorporating computer science, neuroscience, and visual art. The aim of researchers in this field is to determine which factors contribute to the appeal of a work of visual art. At this point it is known that symmetry plays a part in our determination of what we consider beautiful [4]. Beyond that there have been postulations about a variety of line types stimulating the specialized regions of the occipital lobe, and this rudimentary stimulation being one of the first steps in perceiving beautiful art. Clearly object recognition and high level processing, including contextual understanding, play a role. We must understand the perception and characterization of beauty at many cognitive levels before we may automatically generate pleasing creative works. Kurzweil discusses mechanisms which may be used to create appealing art and music procedurally once we have greater insight into human cognition.

2.5 Synesthesia

Synesthesia occurs when several phenomena are perceived in unison. Examples of synesthesia include hearing colors as different tones, seeing sounds as different colors, and seeing numbers in specific colors. There are several hypotheses for why this may occur, proposed to take place on an architectural level or an anatomical level [5]. Some believe the myelin sheath which electrically insulates nerves is too thin at points and allows cross-talk between adjacent nerves. Another study examined mutations in the genes which determine how excess connections in the brain are pruned. Although this perception of one type of information in terms of another happens in a minority of the population, we try to evoke similar experiences for everyone to participate in. In our information driven society we visualize data, or make it audible through sonification. We use surround sound systems in home theatres to understand where events occur spatially, using both our ears and eyes to perceive and correlate this information. At concerts lights are modulated by the music to heighten the experience. We consciously attempt to bring stimulation across sensory boundaries. We attempt to create synthetic synesthesia as a way of increasing the available bandwidth for perceiving information.

2.6 Particle Systems and Vector Fields

Particle systems can be used to simulate and visualize a variety of physically real and surreal phenomena. Particles can be complex entities such as birds, or simplistic entities such as dust particulates. The particles can be moved by simple forces such as gravity, or complex forces such as swarming behavior. Often times if an artist needs to render smoke, fire (Figure 2), fog, or other volumetric effects, particles are used. Loosely

gathered particles may serve as a good approximation of volumetric phenomena with varying density and a lack of defined surface. Though there is no solid surface geometry created between particles, tightly grouped particles will appear as a continuous surface. Density is conveyed through opacity; where many particles co-exist their overlap will be opaque, becoming translucent as the density is reduced.

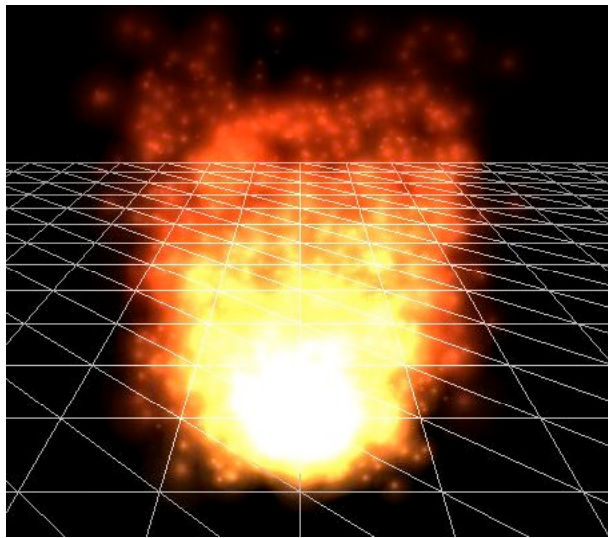


Figure 2. A particle system used to simulate fire

If effects such as fog or smoke must be rendered, one means of moving the particles over time is with vector fields. A vector field may simulate turbulence in the air. At each location in a 3D grid of cells there exists a vector which influences particles by moving them in the direction of the vector with a magnitude based on the particle's proximity to the cell. As particles drift between cells, they are effected by fractional amounts of several cells and essentially interpolate between the vector values. Though the structure of the vector field is simple, interaction of vectors can create some complex

effects such as spirals or convection currents in smoke. The vectors may be created arbitrarily or based on a physical simulation. For example, in a smoke simulation the Y component of the vectors may vary based on the temperature of fire in that region of the XZ plane.

In some cases vector fields representing real physical phenomena such as electromagnetic fields have been visualized and the resultant image presented as information based art [6]. The gratification of the visual representation of scientific data can be used to stir the interest of students and further their understanding of the data. To make information visible also allows individuals to apply their ocular sense to comprehension. Humans have more computational power applied to vision than to the other senses, so it is wise to leverage this. Apart from vector fields used for scientific purposes, vector fields have also been used in an artistic context to guide the path of brush strokes in a painting-like rendering of an image.

2.7 Swarming Behavior

Swarms refer to collections of entities which move together using rules which govern individuals within the group, as well as rules that define interaction with members of the group. This behavior is known as *swarming behavior*. In nature there are many instances of swarming behavior. Fish swim in schools, birds fly in flocks, insects harvest in swarms, deer move in herds, and wolves hunt in packs. A swarming simulation generally takes into account environmental forces acting upon the individuals, forces the individuals are responsible for, and forces causing interaction between the particles. For example, in a swarm being used to represent a flock of birds, the environment may exert

gravitational forces against the birds, the birds will act upon the environment through the locomotion of wing flapping in order to fly towards individual goals, and the simulated birds will stay in some sort of loose formation based on the velocity, direction, and position of the other birds. In a sense the swarm will have a group goal to move towards, which may be an average of the goals of the individuals in the swarm. There may be various rules governing the behavior of an individual in relation to its neighbor. For example, a bird may try to match its orientation and speed to its neighbor, it may try to avoid being too close to neighbors, and it may try to not stray far from its neighbors or the center of mass of the swarm.

In nature swarm behavior may benefit the group. The separation between individuals allows for movement and allows for distributed sensing by the individuals. If an individual moves in accordance with neighbors, it will be benefited when a neighbor senses and moves away from a predator. An individual may sense food and move towards it, and its movement alone may suggest a heading to other swarm members. In terms of increased chances of survival, swarm behavior may have evolved in organisms because it proved beneficial to the population. In the digital realm swarming behavior may be important in terms of simulating a population of organisms, but its contribution to abstract visual art may be less clear. Animation itself is captivating in that it contains movement, as living things do. If the content of the animation is somewhat abstract, such as the movement of geometric shapes, swarming behavior imbues the entities with an organic behavior and a sense of life [7]. We will attempt to make the animations more immersive by alluding to living things and their swarming behavior.

2.8 Genetic Algorithms

Genetic algorithms (GAs) allow for a biologically-inspired breeding process to be applied to inorganic entities. The concept is simple and elegant. Just as the most fit and capable individuals survive and reproduce within a population of individuals in nature, the most desirable designs of a simulated entity will be kept through a breeding process that refines the individuals over many generations. Typically the individuals are randomly initiated for the first generation, then evaluated and bred based on performance over sufficient generations to cause convergence. The GA may also be stopped after a specified number of generations have been evaluated, or after the total computational time reaches some limit, or after a user recognizes some emergent behavior, or some combination of these conditions. The evaluation of an individual is based on the phenotype, or outward behavior of the individual. The evolution of individuals is based on their internal representation, the genotype. Individuals' genotypes are usually represented as binary strings so that genetic operators (selection, crossover, mutation) may be re-used easily between genetic algorithms. Often times it takes multiple bits to express the genes, and this effects the implementation of genetic operators. The general steps for a genetic algorithm are:

Evaluation: Simulate the entities and measure their performance given a fitness function. The output of the fitness function is the fitness per individual. Evaluate all the individuals within the population.

Selection: Based on the fitness values calculated, chose the most fit individuals for breeding so that their characteristics may be retained. The selected individuals will be responsible for creating the next generation of

the population. Individuals will be chosen in proportion to their fitness, with the probability of selection represented as p_{Select} .

Crossover: Once two parents are found, this step determines how much and what specific genetic material comes from each. A crossover point is typically chosen at random from somewhere along the length of the binary string representing the individuals. The new individual will possess bits from parent A prior to the crossover point, and bits from parent B after the crossover point. The probability of a bit being chosen as crossover point is p_{Cross} .

Mutation: Some randomization is involved in order to evaluate close variations of fit individuals. Typically bits are toggled, the probability of each bit being flipped represented as p_{Mut} .

One of the critical factors in determining the successfulness of the GA is specifying the fitness function. The GA will traverse a non-continuous search space in order to find individuals satisfying the fitness function, so the fitness function must be well crafted to allow a GA to work for us. The solution found by the GA will only be as good as the fitness function used to characterize it. In a simple scenario, a GA may be used to find the largest number in a population of numbers, and it would be straightforward to assign a fitness value equal to the numerical value of an individual. In another instance we may want to create an additional instrument part in a musical composition. The fitness function may be based on the amount of harmonization in a progression [8], which may involve the state transitions between consecutive notes, the intervals between simultaneous notes, and a lookup table containing weights derived from grammar rules for a particular style of music. The complexity of a fitness function reflects the

complexity of what characterizes an individual as fit. Some novel fitness functions have been explored in the realms of visual art and music [9], as well as more tangible domains such as architecture [10]. Architecture is a useful domain for study in that it fuses aesthetics with practicalities such as the amount of construction material required for a design and the optimal functional use of volumes.

Each of the genetic operators should be tuned to reflect the problem being addressed [11]. The selection method may be one of several popular variations including roulette wheel, stochastic remainder, tournament selection, or some other selection mechanism created. As its name suggests, roulette wheel selection is very much like treating the population as a roulette wheel where each wedge in the wheel is proportionate to the fitness of an individual within the population. A random number is generated, normalized to the sum of the fitness of all individuals, and the wedge the random number falls upon is the selected individual. Stochastic remainder is very similar. Before the random selections take place, each individual with $pSelect \geq 1$ is selected once, and the new $pSelect$ for that individual is decremented by 1. Tournament selection works by randomly choosing a subset of individuals from the population and selecting the individual with the highest fitness to move on to the crossover step.

There are other selection methods which do not promote diversity. For example, an elitist method will create a new population selecting only the most fit individual. This narrows the search space quickly; only the mutated variants of the elite individual will be considered. Unless $pMut$ is a high value, the random walk in the search space from the elite individual may not come across a more fit individual. Some methods support maintaining diversity. In the case where there are several potential solutions, i.e. several

local maxima in the fitness values, there may be several species we wish to preserve.

Sharing functions address the issue of local maxima help to find *niches* in the population.

It is said sharing functions allow for *speciation*, the development of several distinct subpopulations which evaluate well against the fitness function. A sharing function is a specialized post-processing step which is applied after the initial fitness evaluation but before selection.

The crossover function may be implemented in many ways. A single crossover point may be chosen, or many points may be chosen. An equal number of genes may be chosen from either parent and interleaved. The crossover function may result in asymmetrical contributions from both parents or symmetrical contributions. The crossover points must be chosen deliberately such that they either preserve existing genes or have the opportunity to create new ones. Consider two parents 00000000 and 11111111, each having two 4-bit genes. If we force the crossover point to be only at gene borders, this allows only one point of crossover: the fourth bit. The offspring would be 00001111 if we cross from parent A to parent B at bit position 4. If we allow crossover at any bit position, this may produce an offspring such as 00111111. The initial four bits, 0011, represents a gene not found in either parent. As long as the new gene results in a valid phenotype this may be permissible, but it's possible that we have destroyed a gene at a key position which greatly contributed to the fitness of the parents. Preceding and proceeding genes will be intact, but we've essentially added a mutation by allowing crossover between gene bit bounds. In another case, we may erode fitness by alternating genes from either parent; perhaps several genes in a specific sequence contributed to the parents' fitness.

The mutation step is the most simple genetic operator. At each bit position the chance of the bit being inverted from 0 to 1 or 1 to 0 is $pMut$. Typically $pMut$ is low, around $pMut = .01$. A one percent mutation rate allows for most of the exploration of the solution space to come from random re-combination of existing genetic material, with a slight random drift due to mutation. Just as crossover functions must take into account the associations between genes, mutations may also do so. Instead of the simple bit flipping, a randomly chosen gene could be reassigned to a specific value, or the gene could have its position swapped with a neighboring gene, or the gene and its adjacent genes may all receive new values. With some complex mutation functions the line between mutation and crossover may be faint. The key factor in mutation is creating genes which did not exist at those gene locations within the parents. This allows exploration of the search space beyond the parents' genotypes. If a population converges too quickly, random mutations may be the factor which gets the population out of a local maximum and again moving towards a global maximum.

2.9 Emergent Behavior

Emergent behavior occurs when local interaction propagates to global phenomena. Often the local rules are quite simple but the aggregate result is quite complex. For instance with swarming behavior individuals operate based on their neighbors, but as a whole the group exhibits its own behavior. Cellular automata is another mechanism which may produce emergent behavior. In cellular automata simple cells may change their state based on the state of adjacent cells. Cellular automata gets its name from the arrangement of simple automatons into a cellular, sometimes grid-like

structure. Wolfram documents automata rules by which patterns emerge within a grid of cells. In a simple 2D automata, three contiguous cells are taken as an input and a rule determines the corresponding output. One rule exhibiting emergent behavior is rule 30. Looking at the output where black pixels are 1's and white pixels are 0's, we can see triangle shapes emerging. The triangles are self-similar and exhibit a fractal-like nature. For each pixel in each horizontal line of output a sliding window of three cells was used to compute the binary value. The output is centered below the three cell sliding window.

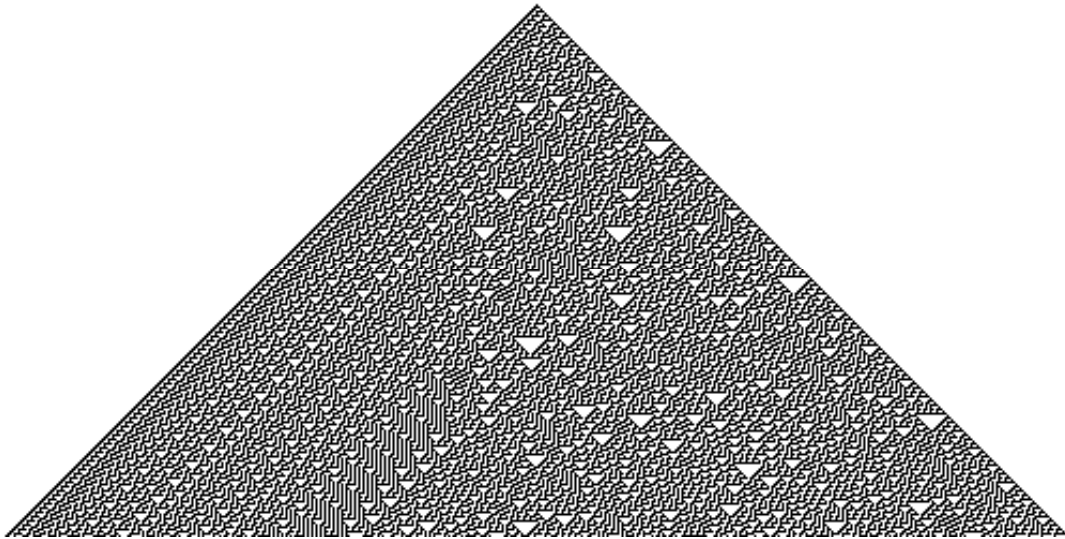


Figure 3. Rule 30

Rule 30 (Figure 3) is stated below in the form *binary 3-tuple input: binary output*

111:	0
110:	0
101:	0
100:	1
011:	1
010:	1
001:	1
000:	0

If we look at the output conditions as a binary string, i.e. 00011110, the decimal equivalent is 30, hence the name rule 30. This is convenient as it gives a unique identifier to the rule and simultaneously encodes the rule.

Wolfram found patterns such as rule 30 in nature (Figure 4) and argued that much of the physics governing the world around us is based on simple discrete calculations that give rise to order [12]. We can look at the human body as a collection of emergent structures: cells with simple logic form tissues, these tissues form organs, and a symphony of organs forms a human. In this sense organisms are built in a bottom-up fashion, the only top-down constraint being the organism's ability to survive and reproduce.



Figure 4. Some patterns found in nature are formed by cellular automata

Just as fractals spawned a niche area of mathematical art and music, cellular automata has been used to generate creative works as well. We used cellular automata to allow harmonies to emerge from a random initial state and was inspired by others who had done related work [13][14]. Because the emergent behavior may manifest over many

iterations rather than instantaneously, a cellular automaton seems somewhat lifelike, as if it's taking time for an otherworldly plant to grow and outstretch its leaves, or as if the automaton is learning to sculpt shapes within the pixels. Whether the mechanism is cellular automata or another mathematical construct, it is the emergent behavior that provides another means with which to make otherwise abstract visual art seem to breathe with life. In order to find emergent behavior in a vast search space we may employ evolutionary techniques such as GAs.

When a computer program's results are beautiful from an aesthetic perspective it may be due to several factors such as complexity, order, ergonomics, and learning [15]. These factors are closely related to emergent behavior. For example, from a blank canvas cellular automata may create an image with geometric structures. Similarly, from a cluttered initial configuration cellular automata may reduce the chaos to geometric structures. Learning may be required to find rules which cause the desired amount of structure to emerge. Another level of learning may take place while determining the target level of structure. As we consider the emergent aesthetic quality of the image we must bear in mind aesthetics from a purely form-based perspective rather than context-based perspective; as forms are constructed by various mechanisms it is more approachable to evaluate them geometrically rather than by how much they resemble items that may be correlated to a viewer's remembered objects. As we attempt to create universally pleasing imagery we may explore the hypothetical existence of common form-driven appreciation of images; it is less likely to find similar context-driven evaluation between viewers.

Chapter 3

Visualizations

3.1 Introduction

In this section we will describe initial approaches to creating the visual content. While our goal was to create a genetic algorithm to enhance the aesthetic value of an animation, it was necessary to first experiment with creating the underlying animation and visualization procedures. We were open minded to several mechanisms for creating dynamic animation including flocking behavior, vector fields

3.2 Initial Particle Swarms

The first visual approach we tried involved moving particles and allowing their paths to be visible. We immediately thought a vector field would provide a good mechanism for guiding the particle movement because it would produce smooth sweeping motions potentially and would allow the particles to have some similarity to their swarm-mates. We seeded the particles along particular shapes, such as a sine wave, to see how that impacted the dispersal pattern. We colored half the particles blue and half red to have two sub-swarms, with one group using a negative coefficient to the weight of the vector field's influence. We applied a gradient as the particle path location grew older.

In many cases we found the confines of the vector field a distraction (Figure 5). In some sense it was an interesting juxtaposition with the fluid shapes created by the particles, but in some regards we would have liked the particles to maintain a fluid appearance. Such an appearance was destroyed when the particles followed the “gutters”, the edges of the vector field.

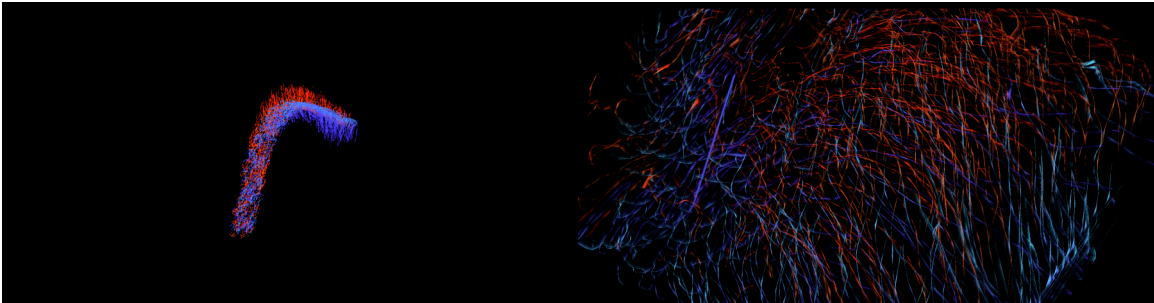


Figure 5. Structure imposed on the swarm through initial position and vector field edges

We then experimented with having the particles pull away from the vector field, back towards the origin. The pulsation back to the origin happened on the most major downbeats of the music, the first beat of each measure (Figure 6). We experimented with contracting the particles by a fixed distance towards the origin and also a fraction of the distance from the origin. As the particles and their paths weren't shaded we desired additional depth cues. We implemented a depth of field (DOF) effect by rotating the scene several times, each time rendering and storing fractional pixel values in the accumulation buffer of the graphics card. For a low quality DOF effect we used seven passes, twenty-five for a smoother effect. In each case the first pass was the original scene angle. The next six passes involve rotating the scene by a number of degrees on the X, Y, and Z axis, then rotating by the inverse amount. Adding the next six frames is

similar, with the angle of rotation halved. The third set of six rotations is similar, with each rotation a quarter of the original rotation amount. We rotated about the origin, allowing objects in the center of the volume to be most crisp, items nearest and furthest being most blurred. We also experimented with animating the point of rotation in order to simulate the effect of a camera changing its focal distance.



Figure 6. DOF rendering and rhythmic swarm contractions

3.3 Image Subtraction Methods

Some experiments involved segmenting the subject in a video sequence and using the subject as a graphic element. As one may observe in Figure 7, the final image required tweaking of the dilation and erosion of the subject's pixels. The original group of pixels was obtained by image subtraction between a frame containing a musician and one which didn't. The background was a cloth photographic backdrop which we left some ripples in. The shadows in the waves of the backdrop allowed for dark areas in which the difference between frames wasn't much, causing regions of the musician's face to be segmented and other areas to be ignored. This strange geometry applied to the face added a pleasing graphic design because it made it less obvious that the musician was playing a violin. We experimented with skipping ahead by several frames or switching

between two video captures of performance of an excerpt of a song. The frame skipping and stream switching were both done such that the visual effect was jarring, but the new video segment would still be synchronized with the song. Since the song was 180 beats per minute (BPM), skipping 40 video frames allowed for variety in the video but consistency in the performance of the string part. The string part performed was intentionally chosen to have consistent bowing across measures even if the notes, the fingering, varied. As the fingers would be either out of frame or indistinct, that was not a detail we worried about.



Figure 7. Segmentation of the subject and several variants on erosion and dilation

After extracting the musician from the background, we affected the background pixels geometrically. We added ripples with several overlapping waves of different amplitudes, each animated over time. The use of environment mapping in OpenGL allowed for the reflections, and rendering with a DOF setting with a low shift angle allowed for some basic anti-aliasing as well as the crispness in the center of the frame that we desired.

3.4 Abstract Video Derived Height Maps

In this experiment we wanted to combine the preceding approaches. We wanted to use video, but in a very abstract sense. We wanted to have something non-objective like the particle swarm, but still allude to something tangible from a person's day to day experience. We took some video with a 1080i HD camera while in Boston and used that content for experimentation. We shot some video using a wide angle lens at the beach just as the water was becoming cold enough to freeze. The textures were beautiful as waves of slush washed ashore, and ice and water and sand intermingled.

Because the features were so fine in the beach footage, we went through some of the other video scenes, including some video of trees, a river, several tunnels, and ponds. We were fascinated by the converging lines of man-made structures and the rich textures found in the water. These macro-features were suitable for manipulation. We experimented with accentuating these features – either strong lines or indistinct splashes of texture – by using the images as height maps (Figure 8). With each pixel corresponding to a color and z component in an OpenGL mesh we created several video sequences. Below one may see the result of the experiment.

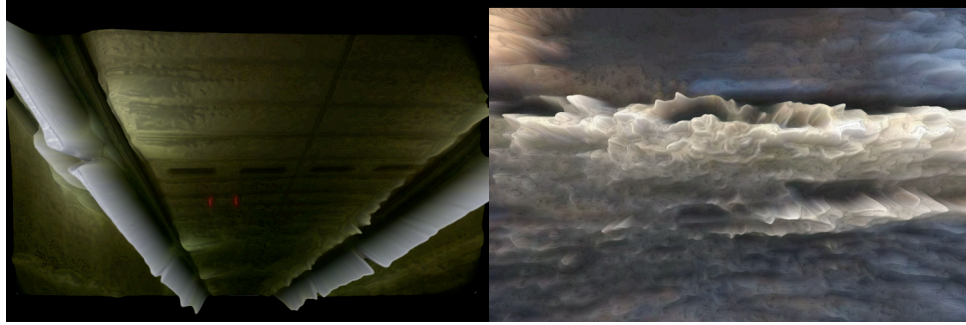


Figure 8. Height map interpretations of video frames

We found the colors and textures of the river especially appealing, and thought it wasn't totally obvious what the source imagery was. It was a goal of ours to obfuscate day-to-day scenery while adding visual interest. With the metallic reflections of the environment map, the video texture was made more intricate. We ran a Gaussian filter on the video frame to tune the amount of noise and the sharpness of the resultant geometry. We found 12 passes with a 3x3 Gaussian kernel produced pleasing results. Before converting the 2D image to the 3D mesh we overlaid some graphics (Figure 9), including text and a grid of squares. We found the squares by themselves added interest when the opacity of the squares was driven by the rhythms in the song. The geometric and color pulsations enhanced the perception of the rhythm.

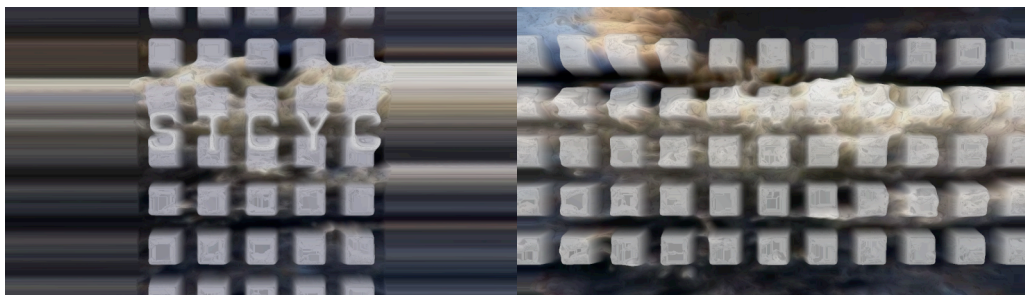


Figure 9. Addition of graphic elements to the height map

3.5 Selection of a GA Modifiable Visualization

As visually pleasing as we thought some of the video driven results were, we decided a scheme that relied on no external aesthetic influence – such as the video footage we created – would be best for experimentation. We also felt the vectors offered such a powerful way to sculpt the visuals, more so than variables such as the amount of erosion, number of blur filter passes, frames to skip etcetera. Those steps could also be applied as post processing to a vector field animation. Performance of the animation mechanism was a concern due to the need to run many iterations with the genetic algorithms. The vector fields could be created at many levels of resolution and still produce high resolution final results due to particles being continuously affected by a weighted combination of nearby vectors, whereas lowering the resolution of an image processing based approach results in more aliased imagery. The number of particles and animation duration are also easily scalable without breaking the feel of the vector field visualization. Perhaps most importantly, the particle motion provides for a fitness function with a quantifiable effect on the resultant aesthetic in terms of form. With that in mind we made a few more variations of the vector field experiment.

Adding environment mapping provided additional depth cues due to the rotation of the objects. Instead of 2D ribbons the particle paths were now more like square tubes following the paths. We created these by finding the normal of the most recent segment of path and a vector from the most recent point to the origin. An orthogonal vector was then found using the first normal and the vector from the most recent point to the origin. Those two vectors were reversed to find the corresponding points on the opposite side of the path, and the triangles were created per path segment to create a tube. Below are

several images showing the effects of environment mapping (Figure 10), face and line rendering to show the geometry (Figure 11), and a comparison of the perceived depth by relying on either the reflectivity created by the vertex normals versus the DOF rendering (Figure 12). We found combining the two made the reflections very soft before the angle was great enough to show DOF, so we used either environment reflections or DOF but not both simultaneously.



Figure 10. Environment mapping

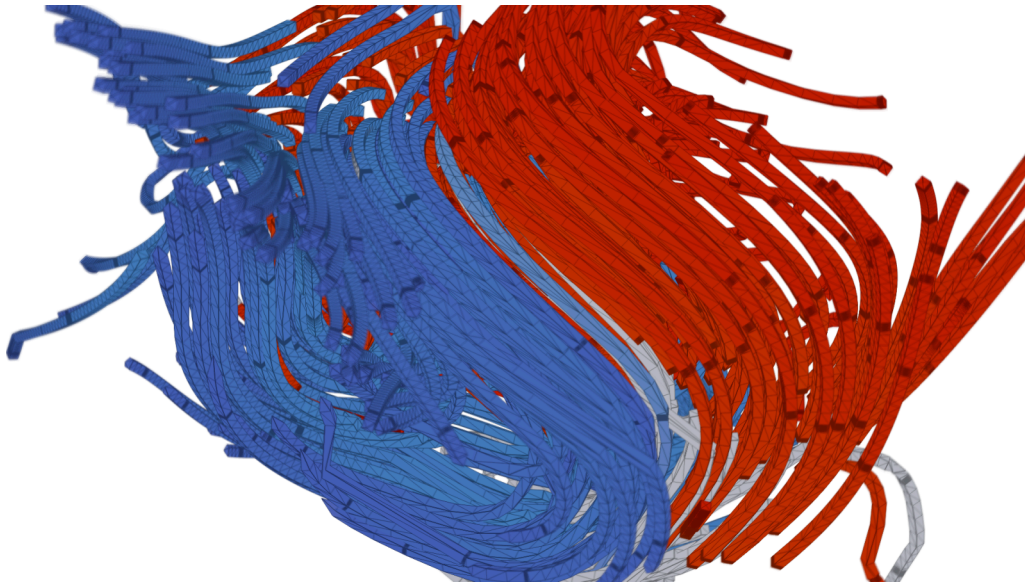


Figure 11. DOF effects with geometry overlay

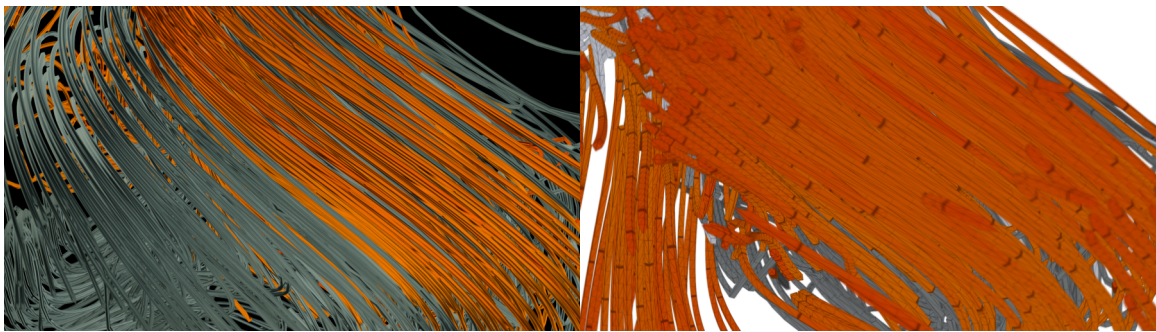


Figure 12. Comparison of depth cues from environment mapping and DOF rendering

To better illustrate the dynamics of the music, we had each swarm's magnitude of motion determined by the magnitude of a musical instrument digital interface (MIDI) event (reduced in ADSR style via a decay factor). Because the MIDI sequence was more complex than the main rhythm we wanted to visualize, we selected the kick drum events to drive one swarm (the grey particles in Figure 13) and the snare drum events to drive another, ignoring other events. The direction of particle motion was still entirely vector field based, but the particle speed in essence was the expression of the instruments.

Because the snare events were more prevalent than kick drum events, the kick drum events were weighted higher to cause both swarms to contribute structure to the swarm based sculpture almost equally. The impulse from the MIDI event effects not only particle speed, but also the color of the particle path. The decay factor is very visible through the slowing of particle movement, but also in the coloration of the particles.

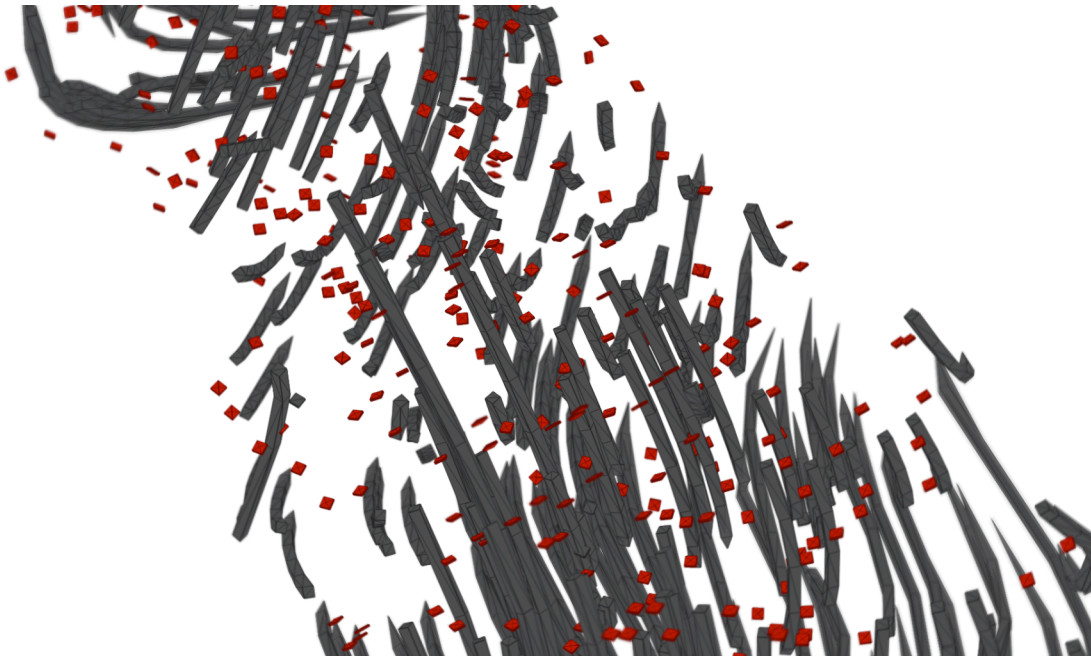


Figure 13. Early stage of synesthetic sculpture construction

When the particles intersected each other we caused the growth to pause, and the path would be rendered white to show the particle was no longer contributing to the structure. One can see the lines in several states of construction (Figure 14), from barely growing to many particles intersecting. While some intersections are between different swarms, the majority of intersections are between members of the same swarms as they condense in vector field edges. In the image sequence shown (in ten frame intervals) the

reddish swarm becomes more orange as the particles age, with only the most recent path segments being red. The grey swarm becomes predominantly white as its members are driven to an edge within the vector field's cube of influence.

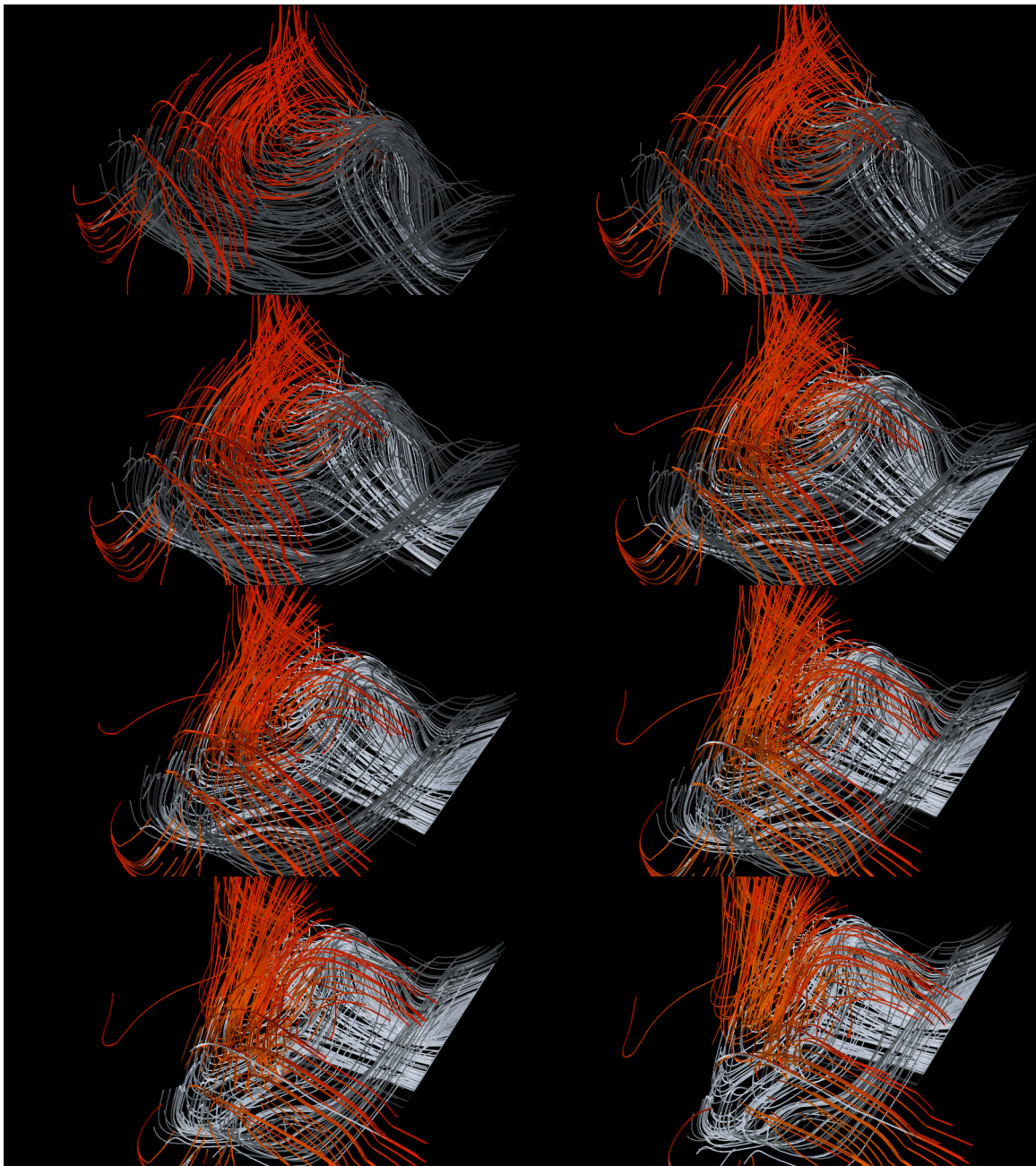


Figure 14. Growth of synesthetic sculpture, 10 frame intervals

We did some experiments with having the particles initially lie in a 2D plane and each swarm be driven by a 2D vector field. We also had each particle increment its height when a particle path from another swarm was encountered (Figure 15), rather than deactivate the particle and render it white. In some cases this caused a particle to cross the path left by another particle, but in the case of a head on collision the two particles increase their Z values until the paths extrude towards the view point. This was an interesting effect but eventually we decided we wanted to have each swarm in a different 2D plane to avoid intersections altogether. Particles from the same swarm tend to weave concentric patterns because of the compression and expansion the vector field applies, so path crossing is not a concern (Figure 16). The slight intersection in tube geometry is not a problem when rendering with the environment mapping, it looks like fluid wires melting together and then diverging again.

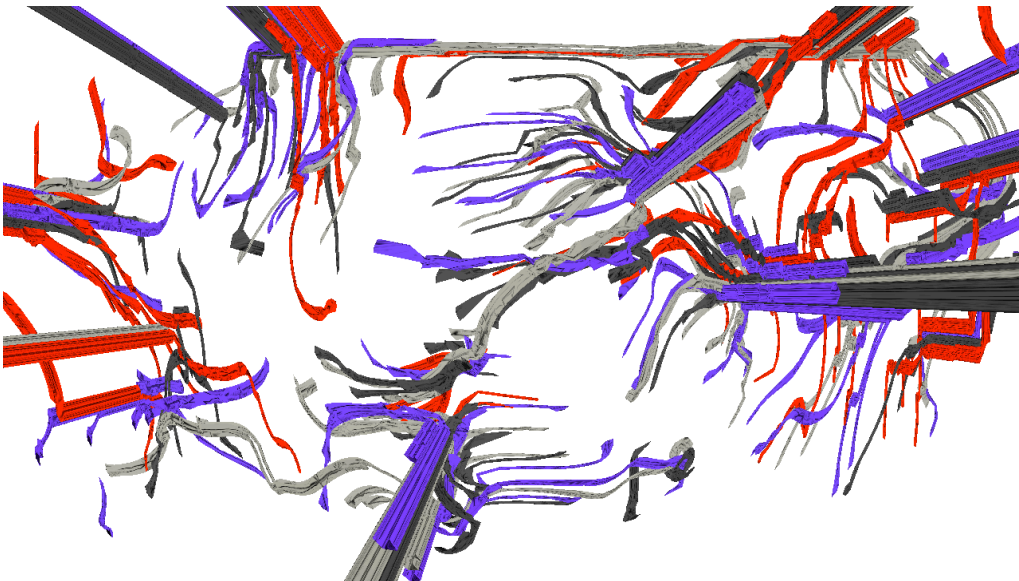


Figure 15. Path crossings causing extrusion

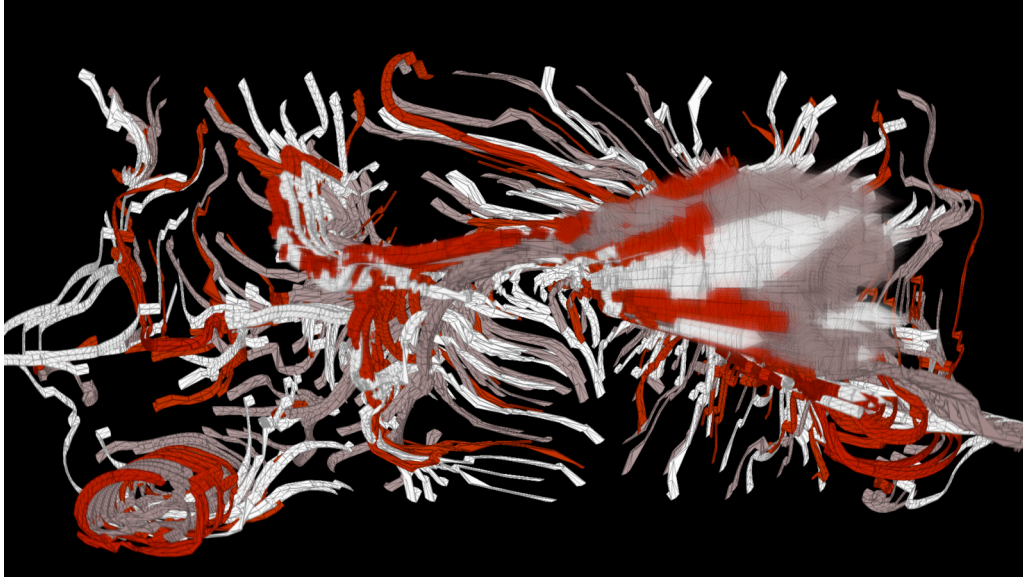


Figure 16. Concentric paths (lower left)

Instead of tubes with equal width and thickness, we caused the thickness to be somewhat less than the width, giving us something of a ribbon effect. This made the paths' position relative to the origin very interesting, causing the ribbon to twist during turns. This twisting is due to the use of the cross product of the vector from the particle to the origin being used to calculate the initial normal used in constructing the tube geometry. After sacrificing a dimension of the vector field by making it a 2D vector field we wanted to animate one more item to lend interest. We then had the particles create the tube geometry using not the origin for the initial normal, but an animated point near the origin. The point is at the origin in the X and Y dimensions, but varies with the MIDI impulse along the Z dimension. This point was different for each swarm since each swarm is driven by different MIDI events, causing the swarms to appear to breathe and pulsate in rhythmic harmony. Lastly, to have some interplay between swarms, each swarm is affected by 100% of its vector field's strength and 10% of each of the other

vector fields. This manifests in patterns of similarity between swarms, as though the swarms were slightly chasing each other. With this new setup we were ready to explore variations on the structures created by the particle swarms.

Chapter 4

Genetic Algorithm Design

4.1 Introduction

The genetic algorithm (GA) functions with the general stages of selection, crossover, and mutation, but with special considerations to account for relatively small population size and two dimensional associativity of the genes. Because we experimented with different variations on the crossover, animation duration, and other aspects, we started with the same initial particle positions and vector field configurations across all executions.

We modified the renderer to allow for saving not just each animation tick, but speeding up the execution of a generation by only creating geometry and raster data for the final animation frame. This final frame “snapshot” was invaluable for assessing the validity of the fitness function and progress made by the GA operators. We also created a utility for tiling the snapshots with overlaid fitness scores in order to see each generation at a glance. The vector field genes for each individual in each population were also output to allow the individual to be imported for rendering of the full animation at a later time. The animation could later be created with more or less particles, or rendered at a different quality (i.e. more DOF angles), or in a different mode (i.e. with environment mapping turned on) using the vector field of best fitness. Below is an ending condition of an individual. The three swarms are grey, orange, and red, with particles which have collided with vector field borders rendered in a lighter version of those colors in order for

the user to observe the relative success of the swarms (Figure 17). In these evolved sequences, the red particles are moved by the kick drum, the grey particles are linked to the snare drum, and the orange particles are moved by the high hat in the drum sequence.

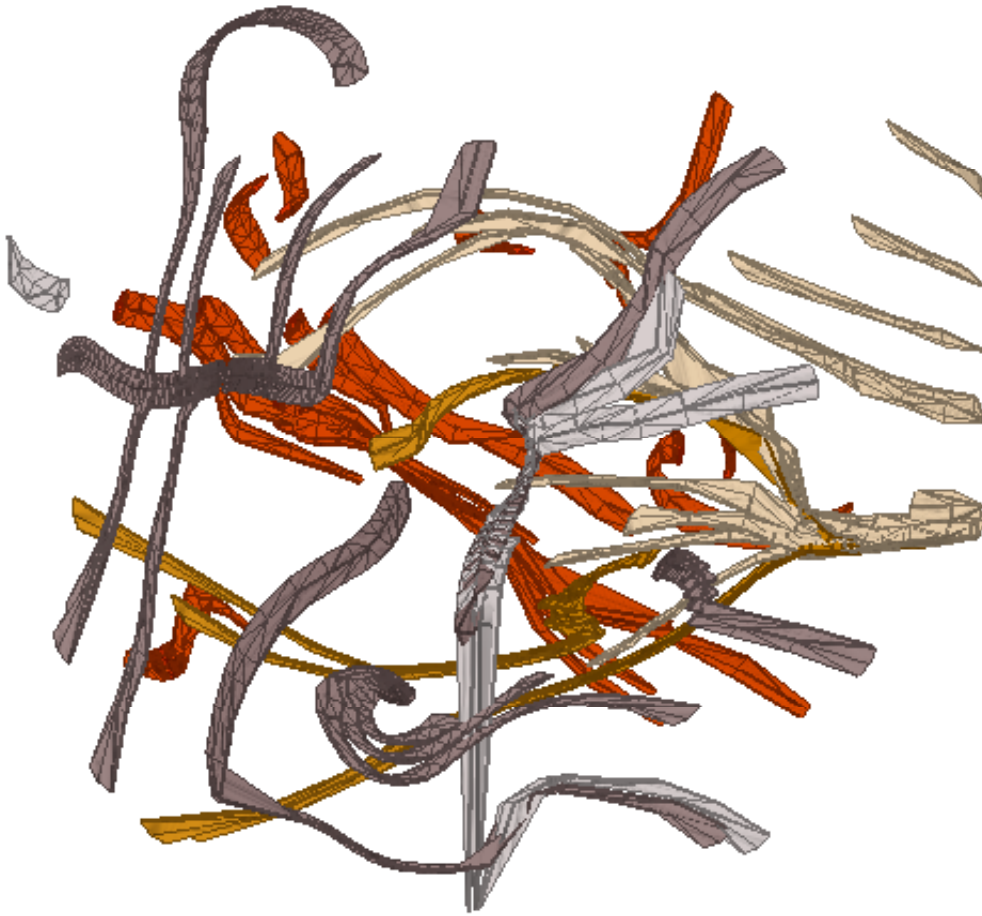


Figure 17. Three swarms corresponding to drum components

4.2 Fitness Function

The behavior we primarily wanted to avoid at this point was having the particles collide with the vector field borders, changing a visual composition of otherwise flowing

organic lines into a composition having sharp 90° edges. We decided there would be two simple aspects to the fitness function. We wanted a fitness function that rewarded not colliding with vector field borders. To that end, any particle which collides with a border is marked. Each particle which is still unmarked at the end of the animation sequence adds (particleCount/10000) to the fitness value. We also didn't want particles to be stationary as a strategy of avoiding borders, so vectors must be one of four normalized vectors $\{1,0\}$ $\{-1,0\}$ $\{0,1\}$ $\{0,-1\}$. Each particle is affected by a weighted combination of vectors, including the elements of other swarms' vector fields, so an additional motivator for particle motion had to be added. At each animation tick the individual is rewarded for speed of the particles, and the fitness value is incremented by the sum of the particles' speed.

4.3 Selection, Crossover, and Mutation

After running a population of 50 individuals over 100 generations we noticed that the generation's fitness could rise and fall dramatically. The population was too small to force preservation of aspects of the previous best individual. We allowed the best individual seen thus far to be copied directly into the new generation, altered only by mutation. We didn't want to flood the new population with the best individual as in an elitist approach, but keeping one copy with subtle mutations amongst a population filled with more radical results of crossover provided good diversity and stability in maximal fitness per generation.

We were concerned about having the vector fields destroyed by copying only particular rows since the genes are accessed in row-major order. To give equal chance to

preservation (or destruction) along horizontal and vertical divisions, we created a two-dimensional crossover operator. The operator chooses cx and cy from zero to the dimension along each axis (in our case, six). The crossover takes the vector field locations where $(x > cx \ \&\& \ y > cy) \parallel (x < cx \ \&\& \ y < cy)$ from one parent, and the remaining locations from the other. This is done per vector field, acknowledging that regions of good particle manipulation in one vector field may be weak in another. This allows for each vector field to potentially improve independent of the other vector fields. The parents, which are chosen via roulette wheel, will not necessarily be equally represented in the offspring. An elite individual was added, chosen from all preceding generations. This is similar to a single tournament style selection where the tournament size is equal to the population size. Though the parents are chosen based on fitness, the ratio of genetic material contribution from each is random with an average of 50%. The final stage in creating individuals for the new generation is mutation. The mutation rate is 5%. The mutations are not uniformly distributed across the vector fields of an individual.

Chapter 5

Genetic Algorithm Implementation

5.1 Introduction

In this section we will outline the software implementation for this project. The software consists of SwarmGA, which contains the particle system with fitness evaluation, genetic operators and image renderer, and a second program, StatsGA, for calculating the statistics and creating the generational snapshots. Both of these programs are written in Java.

5.2 Implementation of SwarmGA

The purpose of SwarmGA.java is two-fold: It must create, evaluate, and breed individuals to explore the search space, and it must render the results so that we may compare fitness to the final animation frame. Once we've found an individual of significant fitness and have chosen to render the full animation with the Java OpenGL bindings (JOGL), SwarmGA may be used with a different render setting to create the animation. In GA mode a batch typically takes six hours to complete on a 2.5 Ghz G5 computer. In animation mode the execution time is typically 2-10 hrs depending on frame resolution and DOF parameters.

The process of SwarmGA, in pseudocode when in GA mode:

```
Initialization
    Create initial population of vector fields from data file
    Create list of initial particle positions from data file
Simulate individuals per animation frame
    Nudge preceding path positions
    Update newest particle position using vector field
    Scale particle motion with MIDI triggered ADSR envelope
    Limit particle motion
    Deactivate individuals which have collided with borders
    Increment fitness by particle speed
Render
    Create geometry based on particle paths
    Render filled triangles
    Render triangle lines to show speed over time
    Save image as JPEG
Evaluate
    Increment fitness based on number of active particles
    Save vector field elements and fitness to text file
GA operators
    Create new population via roulette wheel
    Substitute elite individual
    Return to simulation
```

The process of SwarmGA, in pseudocode when in animation mode:

```
Initialization
    Create initial population of vector fields from data file
    Create list of initial particle positions from data file
Simulate single individual per animation frame
    Nudge preceding path positions
    Update newest particle position using vector field
    Scale particle motion with MIDI triggered ADSR envelope
    Limit particle motion
    Deactivate individuals which have collided with borders
Render
    Create geometry based on particle paths
    Render filled triangles with environment mapping
    Save image as JPEG
```

5.3 Implementation of StatsGA

The purpose of StatsGA.java is to analyze the improvement in fitness over generations. The min, max, and quartiles are calculated and graphed per generation.

Additionally, final frame snapshots are tiled to create an aggregate snapshot of each generation. Between the quantitative analysis of the fitness graph and the qualitative analysis of the generation snapshot we are able to observe how the fitness function alters the phenotypes. StatsGA typically takes less than five minutes to run, depending on the resolution of the final frame images. The process of StatsGA, in pseudocode:

```
Fitness analysis per batch
    Load fitness scores of each individual per generation
    Sort fitness scores per generation
    Normalize data using fitness score of least fit individual
        In generation 1 set to 1.0
Plot graph
    Draw normalized min, quartile 1, median, quartile 3, max
Generational snapshot construction
    Load the generation's final frame images
    Tile in row-major order
    Overlay individual's ID and fitness score
    Save image
```


Chapter 6

Results and Observations

6.1 Introduction

We will present the results of several configurations of the system, with revisions to the genetic operators interspersed. As a preface to the per-generation illustrations, in Figure 20 one may see a general example of an unfit system (generation 1) and a fit system (generation 100). In each generational snapshot, each individual is marked by the individual's index within the populations across all generations, followed by the fitness of the individual.

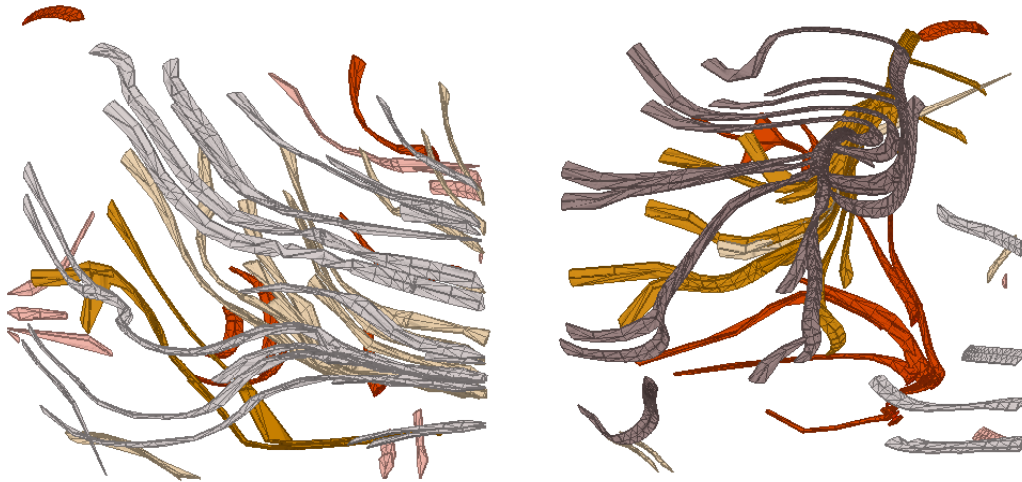


Figure 18. Substantial improvement in fitness between generation 1 (left) and generation 100 (right)

6.2 Initial GA Execution

In this batch we ran the system with a 6x6 vector field for each of three swarms, animating over 40 animation frames. The vector field size and frame count were chosen such that there was a fair chance of particles reaching vector field borders and the GA would have the opportunity to discriminate between good vector fields and poorly performing vector fields. There are 20 particles per swarm. Each generation has 50 individuals. 100 generations were created and executed. Following are generations 1, 50, and 100. One may observe the speciation; different swarms are moving towards fit configurations without necessarily biasing the fitness of the overall system.



Figure 19. Generation 1



Figure 20. Generation 50



Figure 21. Generation 100

6.3 Limited Elitism

In this batch we ran the system with a 6x6 vector field for each of three swarms. Population size is 50, 20 particles per swarm, 100 generations, each individual moving particles over 40 frames as in the first GA execution. The difference with this and the preceding is the addition of retention of the best individual seen thus far in order to stabilize the maximal fitness. One may observe a general trend in the first individual of each generation as that position repeatedly is occupied by the current fitness leader (with mutations). With the large amount of variability in the crossover of each vector field, this elite individual is able to maintain some well-serving genetic material for the population. Depicted below are generations 1, 50, 99, and 100. Of particular note is the similarity of the first individuals in generations 99 and 100. This demonstrates the retention of the bulk of the elite individual's vector field configuration.

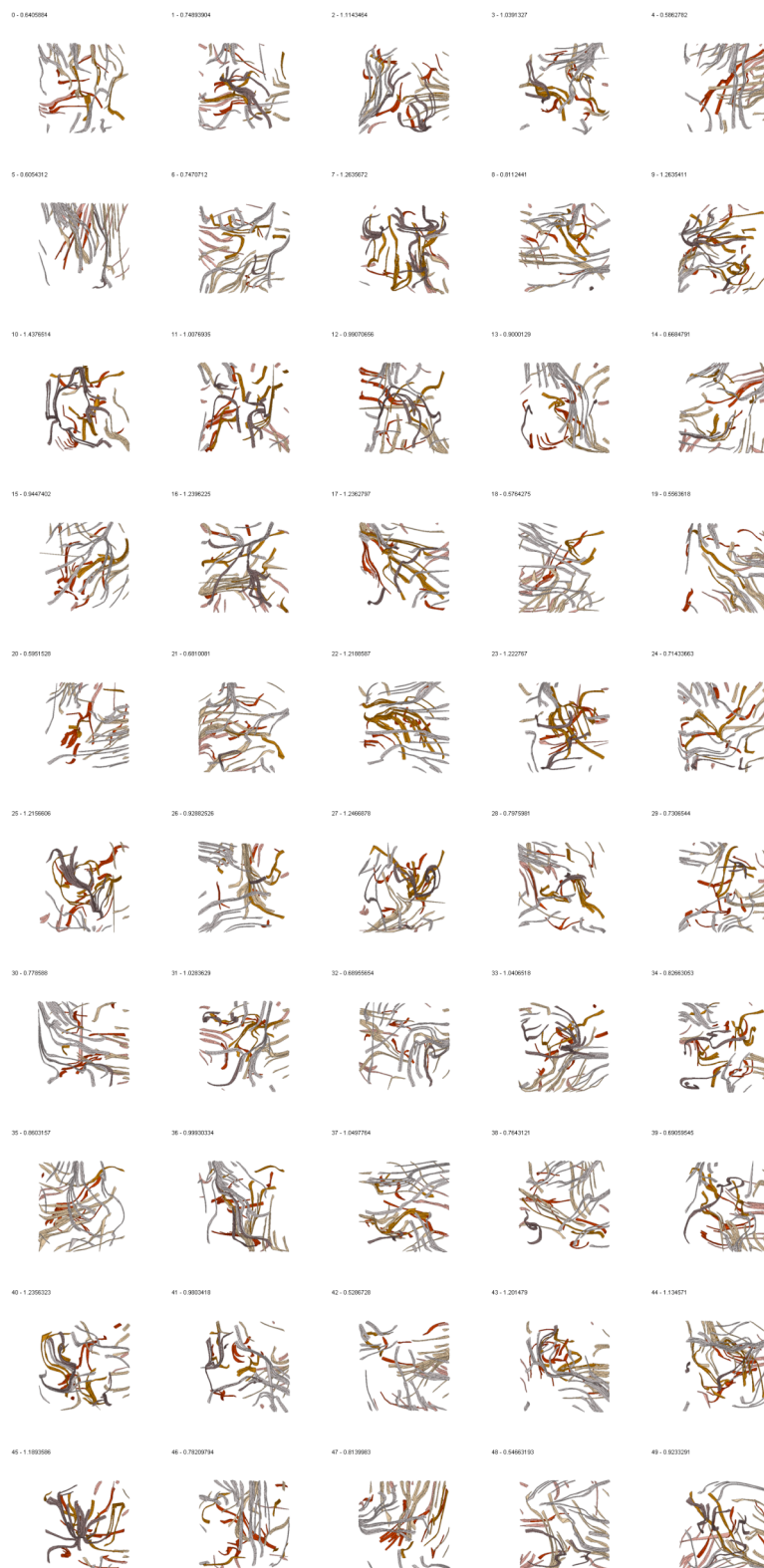


Figure 22. Generation 1 – limited elitism



Figure 23. Generation 50 – limited elitism

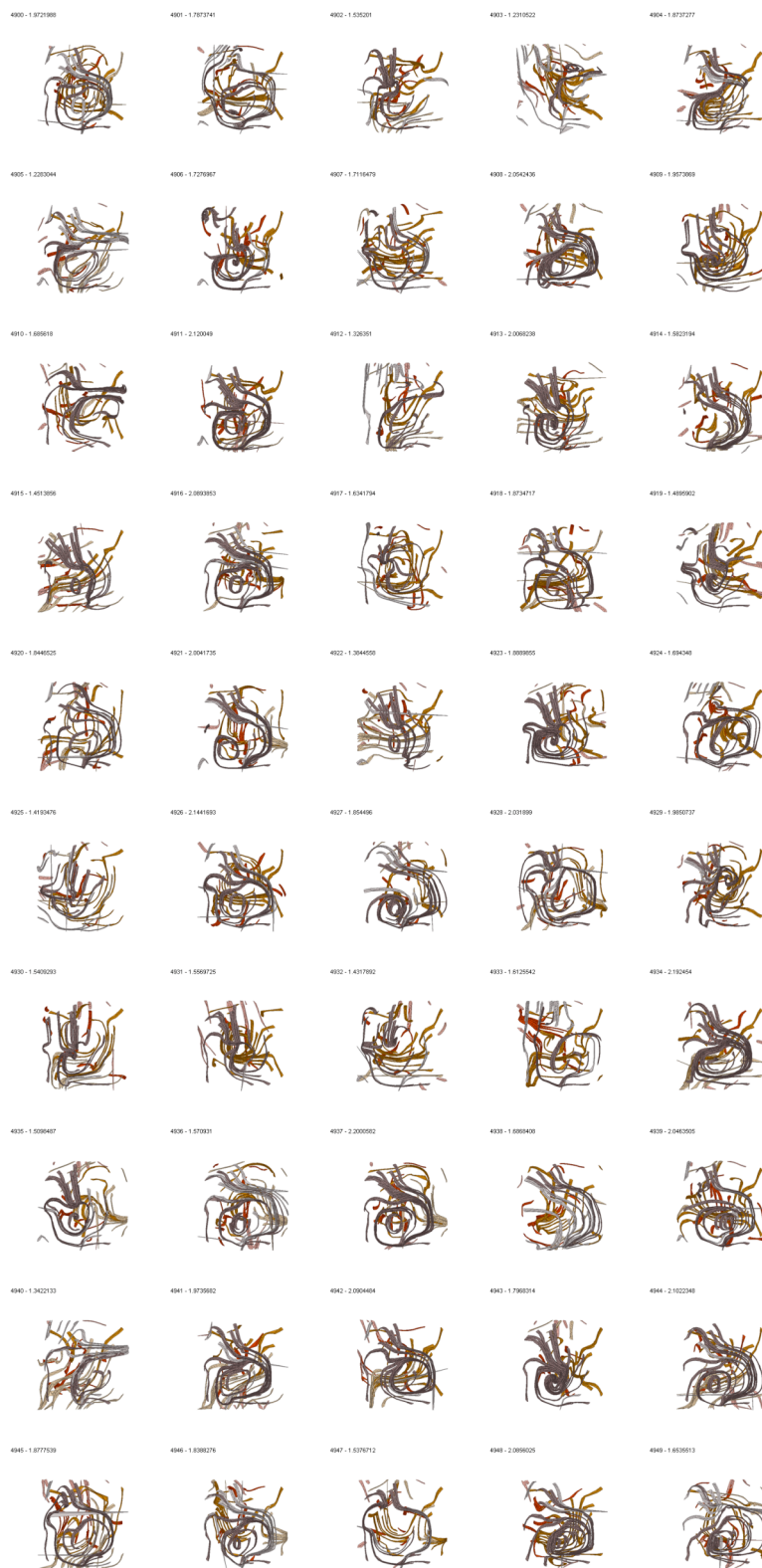


Figure 24. Generation 99 – limited elitism



Figure 25. Generation 100 – limited elitism

6.4 Differing Population Sizes and Generations

In this experiment we structured 10,000 individuals in several ways. Each run was using 40 animation frames, 3 swarms, 20 particles per swarm. What varied was the allocation of population size and number of generations. We ran the software with populations/generations of: 100/100, 50/200, 25/400, and 10/1000. The question we wanted to ask was whether a smaller population size could be offset by additional generations. We also wanted to know if a small population size would limit diversity in solutions too quickly. After normalizing the data, setting the least fit individual in the initial population to fitness value of 1.0, we were able to compare the relative increase in fitness.

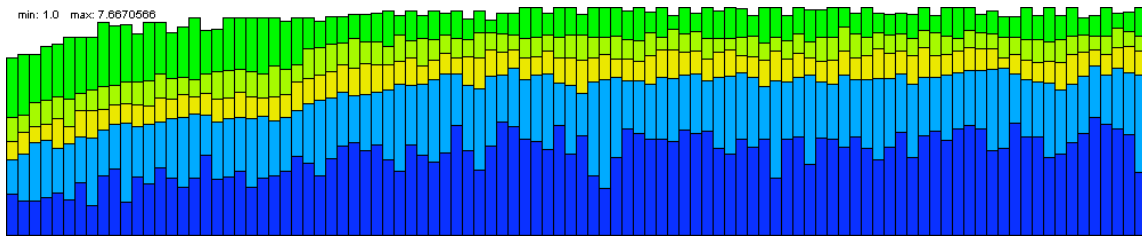


Figure 26. Scenario A – max 7.7

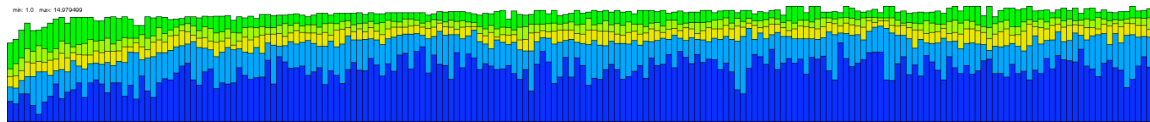


Figure 27. Scenario B – max 14.9

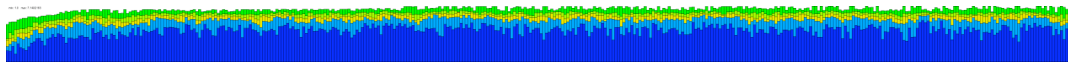


Figure 28. Scenario C – max 7.14

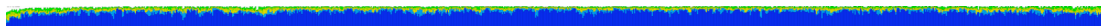


Figure 29. Scenario D – max 4.6

The execution with population size 50, 200 generations (B) showed greater improvement in fitness than the execution with population size 100 (A), indicating the extra computational time used for twice the individuals may not be beneficial. Also, in (B) we see there is little improvement after generation 23, indicating that a large number of generations are not needed to see some significant improvement. In (C) and (D) we notice that although we still allow for a full 10,000 individuals to be explored, the smaller population size causes the elite individual to have too large an impact on the population, and the stochastic remainder method cannot create enough diversity once the population begins to converge. The mutation rate is too low to allow for radical changes to the vector field structure.

6.5 Full Resolution

In this batch we ran the system with a 20x10 vector field for each of three swarms. Population size is 100, 100 particles per swarm, 100 generations, each individual moving particles over 160 frames. The difference with this and the preceding is the more complex vector field, longer animation duration, and broader population. This is most representative of the system needed to generate the final visual content. Using the preceding experiments we knew the limited elitism aided our search, and that the law of diminishing returns made it unnecessary to run an exhaustive amount of generations. In this experiment we put our hypothesis to the test: Is a population size of 50 adequate to allow diversity and significant improvement in fitness? The longer animation duration created a much broader range of fitness values due to the temporal weighting.

6.6 Emergent Behavior

An interesting thing to note is the emergence of different curvatures to prevent the particles from meeting boundaries. This took the form of switchbacks and vortices, always interesting shapes. The concentricity created patterns among members of the same swarm, and the slight weighting of the other vector fields to each swarm in addition to its vector field caused some interplay and co-evolution between vector fields. If one clusters the individuals based on hamming distance, one may see the genetically similar individuals are strikingly similar visually. We would expect a strong genotype-to-phenotype correlation.

From this initial experimentation it seems genetic algorithms can be used to enforce an aesthetic style in a visual composition. As we learn more about what contributes to beauty from a computational aesthetics point of view we will be able to expand upon the existing fitness function. In terms of addressing a need for symmetry for aesthetic reasons the swarms are clearly not symmetrical with their paths, but the center of mass seems to be near the center of the vector field. Though we expected concentric structures due to the vector fields, we did not expect such fluid transitions from groups of particles moving independently to groups of particles forming vortices, then again dispersing. The patterns express movement in both the individuals and the swarms. The patterns in the final images became more pleasing after iterations of the GA caused the vector fields to conform to some basic rules. While we predicted some trends such as vortices would occur we were surprised at how much more appealing later generations appeared. When animations were created from the high fitness individuals, the concentric paths were especially appealing to watch being constructed. The degree of swarming and

total motion in the scene was greater in highly fit individuals, contributing to the quality of the result. In a sense the emergent behavior exhibited by the fittest individuals is greater beauty through the spirals. An aesthetic style emerged through the particular structures that frequently appeared.

Chapter 7

Future Work

We would like to further explore weighting of the fitness contributors and create new contributors. With a suitable number of fitness function parameters we would like to evolve the fitness function used such that the fitness improves the most dramatically within a fixed number of generations. Due to using different contributors to the fitness, the fitness of the first individual in the first generation of each batch would be used as a constant with which to normalize the data. This approach would border on genetic programming but with a limited search space. We would also like to study speciation in relation to this problem. A sharing function would enable us to refine a system with multiple potential swarms that evaluate with good fitness. Further exploration of the crossover operator would help to avoid destroying vector field features that contribute to the fitness.

To improve execution speed we could implement the vector field simulation on the graphics processing unit (GPU). By using the GPU as a stream processor we could alleviate much of the burden on the CPU, allowing successive generations to be evaluated more quickly. In the case of an evolved fitness function we would definitely desire greater performance. Another possibility would be a parallel implementation, allowing individuals in the same population to be evaluated simultaneously on different CPUs. In this case there would be peer-to-peer communication to assign individuals to CPUs.

Rather than multi-thread the program, we could run an instance of the program per CPU core and see dramatic improvement in execution speed.

References

- [1] Campbell, Don (2001). *The Mozart Effect*. New York City, New York, Harper Paperbacks.
- [2] Olsen, Sven, Maxwell, Bruce A., Gooch, Bruce (2005). Interactive Vector Fields for Painterly Rendering, *Graphics Interface 2005 Proceedings*. Canada, The Canadian Human-Computer Communications Society, pp. 241-247.
- [3] Kurzweil, Ray (2000). *The Age of Spiritual Machines*. New York City, New York, Penguin.
- [4] Tyler, Christopher W (2004). The Human Expression of Symmetry: Art and Neuroscience.
- [5] Hubbard, Edward M., Ramachandran, V.S. (2005). Neurocognitive Mechanisms of Synesthesia.
- [6] Thomson, Elizabeth (2005). "Science becomes art in 'Weird Fields,'" The MIT Press.
- [7] Johnson, Adrian, Semwal, S.K. (2004). Music as an Input Device, *IEEEVR 2004 Proceedings, Beyond Wand and Glove Workshop*. Chicago, Illinois, April 2004.
- [8] Johnson, Adrian (2006). Complexity and Music Generation, *GECCO 2006 Proceedings, Graduate Student Workshop*. Seattle, Washington, July 2006.
- [9] Johnson, Colin G., Romero Cardalda, Juan J. (2002). Genetic Algorithms in Visual Art and Music. *Leonardo*, Volume 25, Number 4, August 2002, p 364.
- [10] Chouchoulas, Orestes (2000). Shape Evolution: An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms.

- [11] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. USA, Addison Wesley.
- [12] Wolfram, S. (2002). *A New Kind of Science*. Canada, Wolfram Media.
- [13] Bilotta, E., Pantaro, P. and Talarico, V (August 2000). Synthetic harmonies: an approach to musical semiosis by means of cellular automata. In M.A. Bedau, J.S. McCaskill, N.H. Packard and S. Rasmussen, editors, *Artificial Life VII*. Cambridge, Massachusetts, The MIT Press.
- [14] Maeda, John (2004). *Creative Code*. London, Thames and Hudson.
- [15] Hoenig, Florian (2005). Defining Computational Aesthetics. *Computational Aesthetics 2005 Proceedings*. Germany, The Eurographics Association, pp. 13-18.

Appendix A

Generational Snapshots

In this section we will provide additional generational snapshots to visually describe the differences between generations. These populations were taken from a batch using population size 100 and 100 generations. Every 10th generation is depicted. Preceding the generational snapshots is a graph of the overall performance change of the population over the generations. Despite slowing improvement beyond generation 15 it's of interest to watch the individuals with high fitness change and refine the paths of the particles.

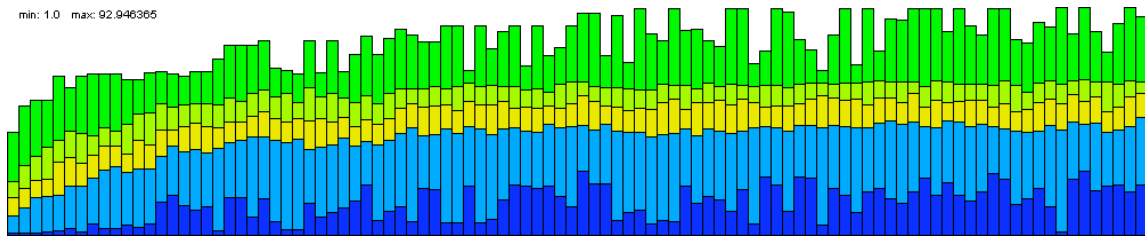


Figure A0. Fitness per generation



Figure A1. Generation 0



Figure A2. Generation 10



Figure A3. Generation 20



Figure A4. Generation 30



Figure A5. Generation 40



Figure A1. Generation 50

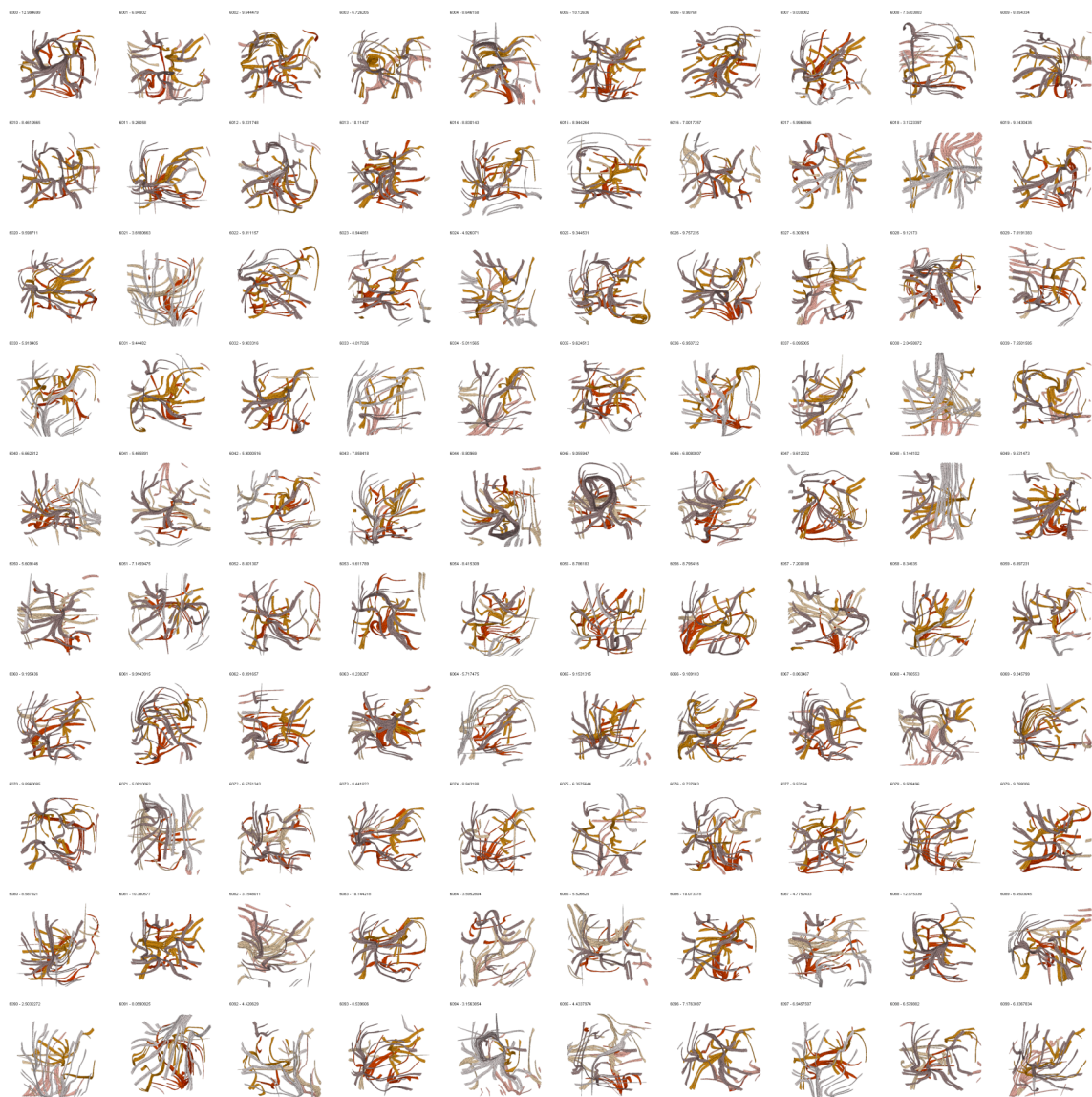


Figure A7. Generation 60



Figure A8. Generation 70

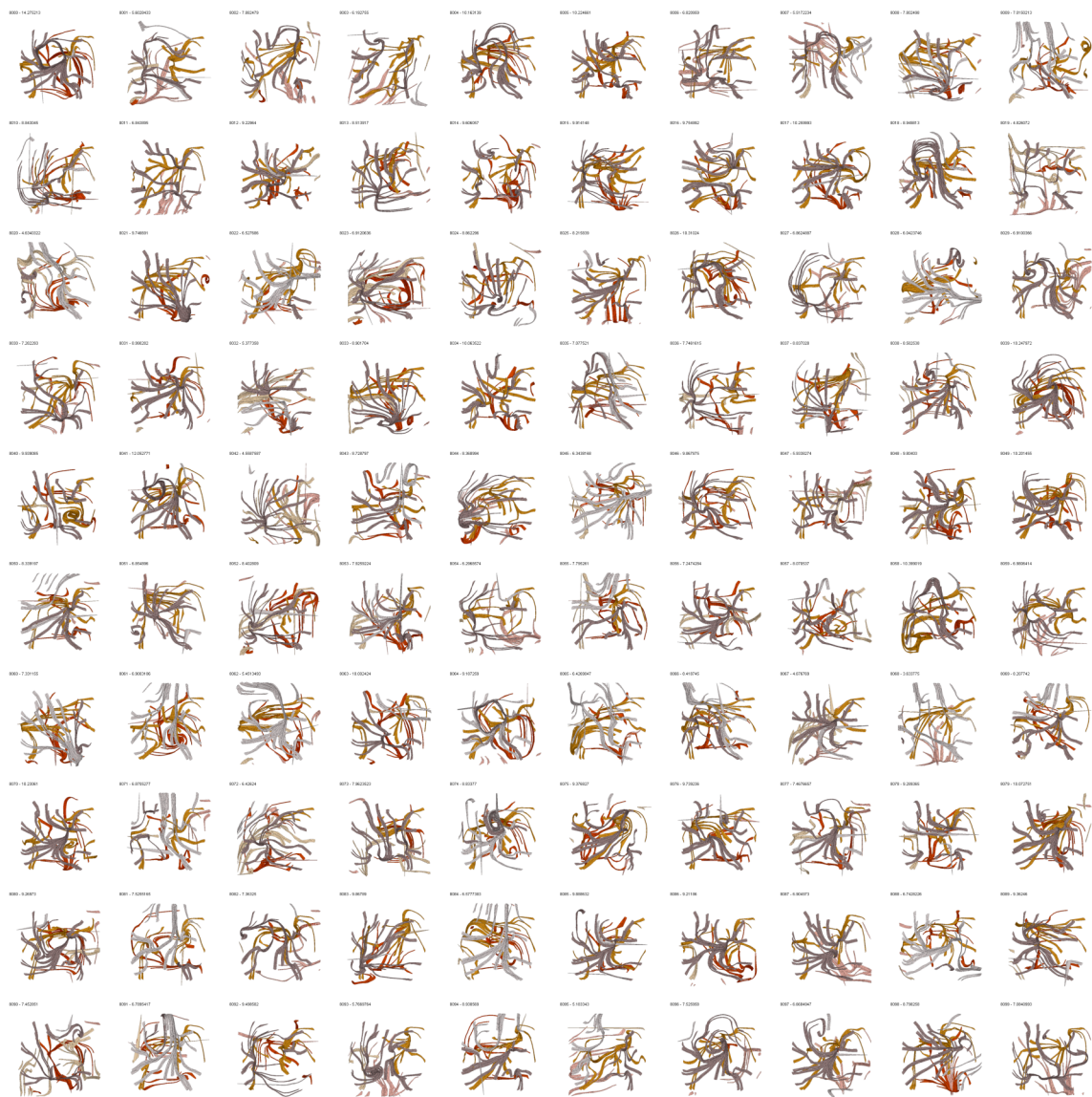


Figure A9. Generation 80

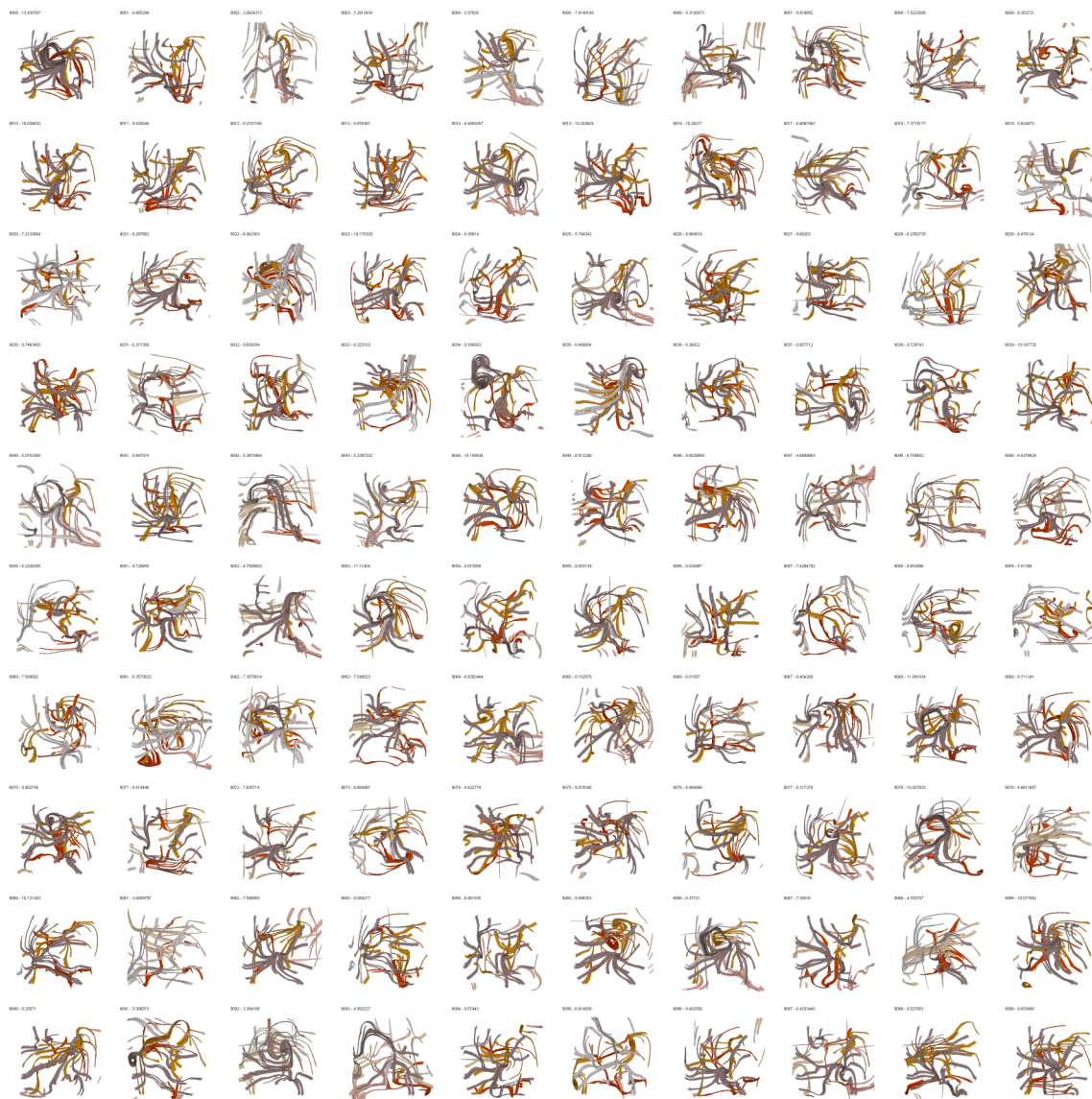


Figure A10. Generation 90



Figure A11. Generation 99