GAUGUIN: Generating Art Using Genetic Algorithms and User Input Naturally

Thomas Cook
Colby College
7530 Mayflower Hill
Waterville, ME 04901
thomas.cook@colby.edu

ABSTRACT

This paper outlines an undergraduate research project demonstrating an application of evolutionary computation in the context of computer art. The project combines the visual impact of modern computer graphics with the computational power of genetic algorithms. GAUGUIN allows the user to become a creator of art, without requiring any technical or artistic training. By using an intuitive and easily comprehensible process like evolution to create the composition, all the user needs to do is evaluate a number of possible "solutions", which trains the system to recognize his or her specific taste. The act of evaluating and scoring is inherent in all of us; this project simply takes advantage of that behavior in a creative way.

Categories and Subject Descriptors
J.5 [**Arts & Humanities**]: Fine Arts.

General Terms
Algorithms, Experimentation, Human Factors.

Keywords
Computer art, evolutionary art, genetic algorithms, graphics, OpenGL, GAUL, interactive art.

1. INTRODUCTION

1.1 Motivation

For the most part, computer art is a highly refined production by the artist. While the user's experience of art may be interactive, he rarely participates in the creative process. Similarly, there are many excellent tools that an artist can use to create impressive computer-based art. However, these tools require a high degree of both technical sophistication and artistic ability.

Genetic algorithms are a powerful computational tool for a wide variety of applications. Some of their greatest benefits include their adaptability, and how they can be used with anything that can be parameterized. Yet another advantage is that they operate based on comprehensible principles separate from the realm of computer science. Every high school graduate understands the fundamental principles of evolution, which makes them able to use an evolutionary system like GAUGUIN.

1.2 Project Goals

1.2.1 Primary Objective

The aim of this project was to present the user with a simple means of producing art that is aesthetically pleasing to them—regardless of their previous technical and artistic training. The user only needs to decide how positively he or she responds to the "solutions" proposed by the system.

1.2.2 Constraints

While each person has a different idea of what constitutes an attractive image, allowing too much variety would make it difficult for the user to effectively compare the images the system presents to them. The intuition for this is that it is much easier to be critical of differences between two similar objects than two objects than have nothing in common.

1.2.3 Style

To limit the range of outcomes without compromising artistic integrity, all solutions are (very) loosely based on an established style of art called Suprematism. This Russian avant-garde movement from the early 20th century is highly geometric, and thus, well-suited for simple computer graphics. A secondary goal of this project was translating the Suprematist style from canvas into three-dimensional computer graphics, and to enhance it with interactivity.

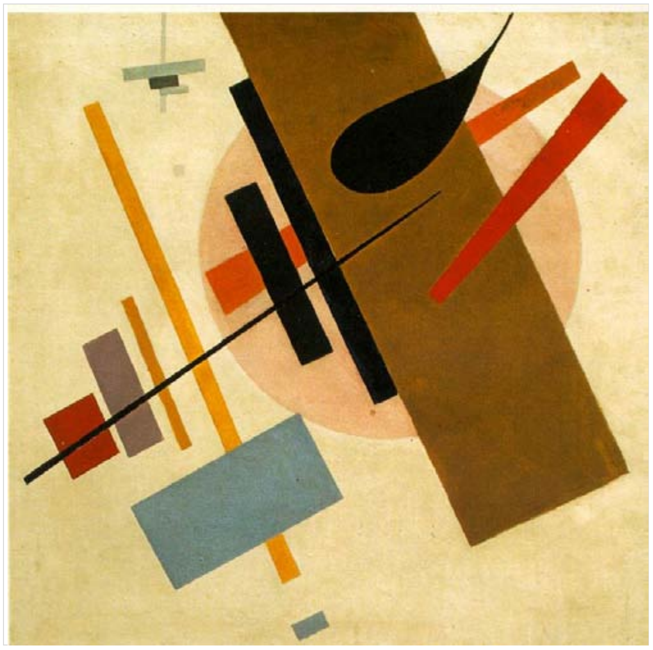

Figure 1. Malevich, *Suprematism Muzeul de Artă*, 1916

2. BACKGROUND

Though a precise definition of art is difficult if not impossible, it is not unreasonable to adopt an "I'd recognize it if I saw it" approach. Most people can agree that art has both a form, which stimulates the senses, and a function, which stimulates the mind. This provides a framework to indentify and compare various works in the field.

2.1 Computer Art

Some of the most cutting-edge ideas and techniques in computer-based fine art can be seen at the San Francisco Museum of Modern Art's exhibit entitled *010101: Art in Technological Times*. [1] These highly creative works are accompanied by an artist statement, detailing their intent. For example, one web project by Erik Adigard called *Timelocator* claims to, "explore the notion of local and remote time." It accomplishes this by displaying the timestamps from when a viewer first opens the page. The background color changes over time from light to dark in sync with the sun at the host server's location. Erik Adigard is a well-established professional artist, like all the people whose works are shown in this exhibit. Together, these artists provide a representative sampling of contemporary computer art.

2.2 Evolutionary Art

Within the broad context of computer art, an entire genre bases itself on the process of evolution. Some works use a genetic algorithm to create finished products that are then presented to the viewer. These types of works frequently make use of the fractal, a recursive shape that is well suited to evolution. Others, like GenTree, utilize user input to evolve realistic looking representations of natural objects, such as trees. [2] GenTree is an excellent example of a system that established clear goals for what it is designed to produce, but still allows flexibility in determining what sort of tree the user like best. Open-ended systems that allow users to decide what it is they hope to achieve without limitation are less common. One example of such a system is Kandid. [3] This program allows the user to select from a number of different styles, color spaces, and other variables, and then evolves images in that style that the user scores. While it is not truly "open-ended" it does offer such a diversity of styles that it is possible to create almost anything with it.

3. SYSTEM DESIGN

3.1 Overview

3.1.1 OpenGL

Since the nature of this project is primarily graphical, deciding which graphics library to use was a an important design concern. OpenGL was selected based on its community support and the amount of documentation available. While capable of highly advanced three-dimensional graphics, OpenGL has a core of simple specifications that permit novice programmers to create graphics with ease. Additionally, a number of libraries like GLU and GLUT extend the functionality of OpenGL by providing simplified interfaces for advanced graphics operations like quadrics and texture-mapping. [4]

3.1.2 GAUL

Another important aspect of this project is the evolutionary system. GAUL, or Genetic Algorithm Utility Library, is an open-source programming library designed to "assist in the development of code that requires evolutionary algorithms." [5] GUAL features a highly developed set of functions and data structures for setting up an evolutionary system, while offering a very simple and intuitive interface that allows for quick integration.

3.2 Graphics

3.2.1 Primitives

GAUGUIN defines a solution as a blank canvas that contains a dozen "primitives". A primitive is a data structure that defines a shape and all its attributes. Each primitive contains a series of integers to define these characteristics. Permitted shapes are quadrilaterals, triangles or circles. Other values include the coordinates of the vertices, the color, and alpha (transparency). Having shapes appear in a logical and discernable pattern adds visual interest, so there is a value that determines if the shape will be repeated one or more times along an axis.

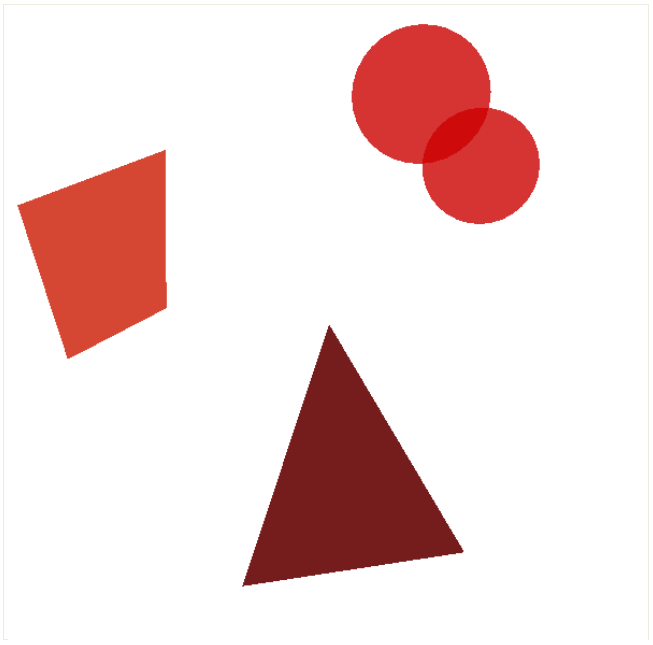

Figure 2- GAUGUIN Primitives

3.2.2 Perspective

Users have the ability to manipulate the point of view in a solution to find the perspective they like best. This effect is controlled simply and intuitively by clicking and dragging the mouse cursor around in the window. The point of view is treated as an integral part of the solution, so whenever this value changes, it is stored along with all of the primitives in the chromosome.

3.2.3 Color

While a variety of hues (colors) add interest to a composition, arbitrary hues are more likely to clash than complement one another. However, giving users flexibility with regards to color would allow them to create a more pleasing composition than simply fixing the palette in advance. To this end, the user is first prompted to choose hues that will form their palette. These colors are permuted, but only within a limited range, ensuring a reasonably consistent look.

3.3 GA System

3.3.1 Evolution

By default, GAUL sets up an initial population, and then proceeds to run an entire life cycle with that population. While this behavior is acceptable with a "static" fitness function, the user is actually the fitness function in GAUGUIN. In order to overcome this obstacle, two changes were made to the GAUL source. First, a function was added that performs all the GA optimization operations to a single generation, rather than a complete life-cycle. This function performs three basic operations on the population: crossover, mutation, and survival of the fittest. Secondly, the default fitness function was removed; instead, the user's input directly affects the entity's score in the chromosome. This allows the user to treat each generation as discrete, reviewing each solution as many times as they desire, and then decide when to move on to the next generation.

3.3.2 Chromosome

Each entity is composed of a number of chromosomes equal to the maximum number of shapes in the composition. This maximum is usually set around ten to prevent the viewer from being overwhelmed by too many overlapping shapes. Each chromosome defines a shape with sixteen alleles. Each allele is an integer with a value from one to ten. Appropriate conversions are performed where appropriate, such as converting the alleles for color to floats. For simplicity, each chromosome has the same allele structure, even though each primitive takes different parameters. This means that some alleles are ignored for certain primitives. For example, the circle requires three coordinates and a diameter, so it ignores alleles [5] through [12].

Figure 3- GA String

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Type	X_1	Y_1	Z_1	X_2 or Diameter	Y_2	Z_2	X_3	Y_3	Z_3

10	**11**	**12**	**13**	**14**	**15**	**16**
X_4	Y_4	Z_4	ΔRed	ΔGreen	ΔBlue	ΔAlpha

3.3.3 GA Parameters

GAUL allows the customization of evolutionary systems through a number of different parameters. The population size determines how many solutions the user will score before moving on to the next generation; for GAUGUIN the population is usually set around ten. This size ensures a reasonably diverse population while acknowledging that evaluating too many solutions without demonstrable progress quickly becomes tiresome for many users. While GAUL supports both Baldwin and Lamarkian evolution, this project uses a simple Darwinian evolution strategy, and allow a single parent with the highest fitness to pass to the next generation. The single parent allows a user to become attached to a particular composition, and evaluate other solutions in relation to it, without too many solutions becoming repetitive. GAUGUIN uses pairwise tournament selection to choose members of the population for crossover and mutation, which occur at rates of 60% and 30% respectively.

4. RESULTS

4.1 Hypothetical Use Case

Sally is a 14-year old girl with a penchant for mischief. Invited to test drive this system, she sits down in front of the screen. Prompted to pick a color, she decides on her favorite, green. She is then presented with an image, along with instructions to press one button repeatedly if she likes an image, and a different button if she doesn't. In Figure 4, sally likes the image on the left the best. Giving it the highest score, Sally repeats the process for four generations, ending up with the image on the right. Satisfied with her work, Sally leaves happy.

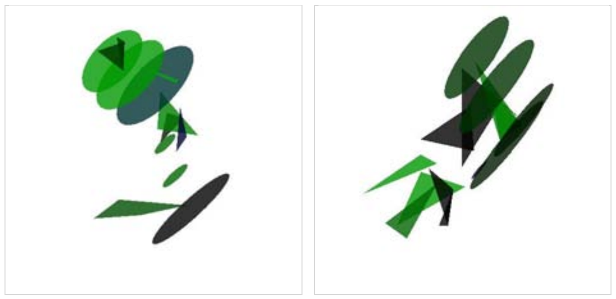

Figure 4- Use Case Compositions

4.2 Limitations

While GAUGUIN offers a unique way of creating visually stimulating images, it trades ease of use flexibility in many respects. The types of images that can be created are limited to a specific style, and a single palette. While the images can be very diverse in the first several generations, under consistent scoring conditions they can eventually become very similar. At this point, the images begin to converge, as seen in Figure 5.

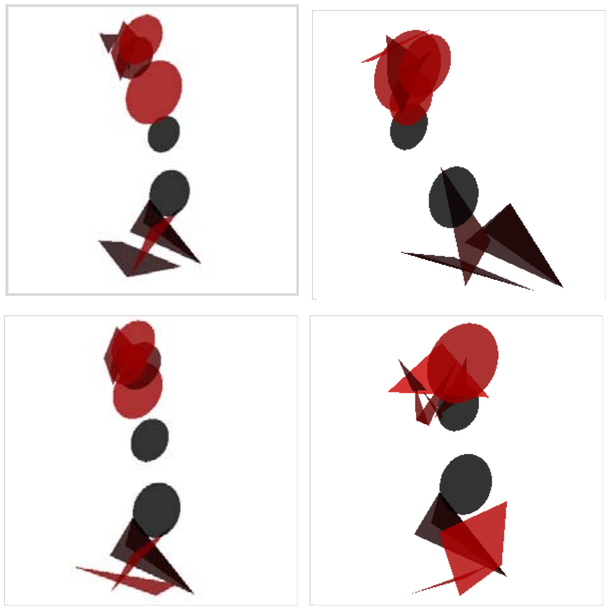

Figure 5- Population Approaching Convergence

The appearance of convergence is both a positive and a negative. While it means that the system no longer presents a diversity of solutions, it does allow a finer grain of discrimination among compositions. Additionally, it serves to provide a sense of closure for the user. GAUGUIN does not have any built-in termination criteria; it is up to the user to decide when they are satisfied. While this behavior makes the system more flexible, it can be frustrating for people expecting a more conclusive finale. Convergence is a natural ending point, as it becomes virtually impossible to achieve significant results thereafter.

5. Conclusions

5.1 Success

GAUGUIN has been successful in presenting the user with a simple and intuitive interface for generating visually stimulating compositions. While the controls are simple enough for children to understand, the computational horsepower provided by the GA system allows a sophisticated method of optimization. While the images created by GAUGUIN bear only a passing resemblance to the Suprematist works which motivate them, they do a good job of taking that style into three dimensions. The ability to view a composition from any angle adds a completely new dynamic.

5.2 Future Work

Like any project, there are a number of aspects of GAUGUIN that could use refinement. The graphics are interesting, but can be very repetitive. It would be nice to add some variety by allowing multiple colors in the palette, as well as adding texture to the background so that the shapes seem to “pop” out at the viewer less. It would also be interesting to constrict the shapes more, so that they line up more on an axis. On the evolution side, a better approach would tailor each chromosome to the data it contains. For example, it would be better to have a float with a constricted range for the color, and an integer from one to four for the shape. Another possibility would be to implement some degree of eugenics, where the user would be able to manually edit the composition throughout the course of evolution. This feature is seen in previous works like GenTree [2]. While this would make the system more complicated to use, it would dramatically increase its flexibility and the spectrum of its output.

6. ACKNOWLEDGMENTS

Thanks to Clare Bates Congdon for her support and advice, and the open source community for providing the code base without which this project would not be possible.

7. REFERENCES

[1] *010101: Art in Technological Time,* San Francisco Museum of Modern Art. http://010101.sfmoma.org/

[2] C. B. Congdon and R. H. Mazza III, "GenTree: An Interactive Genetic Algorithms System for Designing 3D Polygonal Tree Models" GECCO-2003, Chicago, IL, July 2003.

[3] Jourdan, Thomas. Kandid. http://kandid.sourceforge.net/

[4] http://www.opengl.org/resources/libraries/

[5] Adock, Stewart. Genetic Algorithm Utility Library. http://gaul.sourcegorge.net