BRIDGES
Mathematical Connections
in Art, Music, and Science

New Directions for Evolving Expressions

Gary R. Greenfield
Department of Mathematics & Computer Science
University of Richmond, Richmond, VA 23173
E-mail: ggreenfi@richmond.edu

Abstract

In trying to improve upon Sims' art-by-choice technique known as "evolving expressions," artists have taken up the challenge of developing second generation implementations incorporating additional features, modifications, and controls. In this paper we provide a context for this movement, and present a model which both simplifies and refines the original technique. Technical considerations include the role of color space, tools for exploring image space, and rendering issues.

1. Introduction

Biologically inspired computer art techniques, relying on the Genetic Algorithm [5] for search and optimization, have led to the emergence of what Manovich calls "art by choice" [4]. By exploiting increased computer processing capability, the art-by-choice paradigm not only demands new algorithms for searching non-traditional, non-linear spaces on the basis of aesthetics, it also raises questions about the roles of image processing and color management, as well as rendering issues such as lighting and transparency. We should note at the outset that the methods we shall consider in this paper have thus far only been used for creating abstract art, and that final images we shall discuss have been "evolved" according to *subjective* aesthetics. The mathematical notion that is fundamental to this paper is that a work of computer art — an image that is rendered on a computer screen — is an *element* selected from a set of image *points*, and that set of image points comprises an image *space*.

2. Background

The origins of guiding a search through an image space for the purpose of making computer generated art may be traced to computer artist Mark Wilson [10]. At this nascent stage in the development of art-by-choice, Wilson's techniques for stringing together drawing primitives via subprogram calls, and substituting pseudorandomly generated numerical values for formal arguments, represented an artistic exploration of a very simple image space. One might say that this image space, whose image points are elementary drawing programs, is artistically *linear* in the sense that perturbation of the random numbers used as formal arguments or shufflings within the subprogram calling hierarchy produce *expected* results, whence searching the space is possible on intuitive grounds. Indeed, the challenge when using such a scheme is to design "interesting" drawing primitives and "ingenious" hierarchies of calling sequences for invoking those primitives.

A more sophisticated approach to exploring an image space, albeit a space still based on simple drawing primitives, is due to Richard Dawkins [2]. It is no accident that Dawkins' techniques were introduced at a time marking the start of the explosive growth in Artificial Life research. In Dawkins' model, a *genotype* rather than a programmer/artist assumes the responsibility for overseeing the order and frequency with which the drawing primitives will be invoked. The new themes are: (1) representation and encoding of each image point, which is still an elementary drawing program, of image space as a genotype, (2) mutation operators for genotypes, and (3) selection and rendering of a small set of closely related genotypes. Dawkins also made advances in organizing the visual display and creating the user interface for examining *phenotypes* rendered from the genotypes encountered during exploration of image space. Now, perhaps we can better understand why we usually think of image space as being highly *non-linear*. There is often a visual disparity between phenotypes arising from "related" genotypes, and quite possibly the notion of related genotypes is confused or lacks a quantitative measure.

Better publicized than the methods of either Wilson or Dawkins, are the procedural methods of artist William Latham [8], who with the computer expertise of Steven Todd developed the "Mutator" system to create three-dimensional, texture-mapped and ray-traced images on a supercomputer. A noteworthy technical feature of this system is a user interface that allows one to explore the multi-dimensional image space under the guidance of an expert system [9]. For Latham, a genotype is a vector in a twenty-four dimensional real vector space. Though image space is highly non-linear in the sense discussed above, the Latham-Todd solution to the search dilemma in this space is "steering." A record of previous movement within the space helps predict distance and direction for future exploration. If the previous image visited is associated with the genotype afforded by the vector $\vec{g}$, and the current image has genotype afforded by $\vec{n}$, then steering takes place in the direction $\vec{s} = \vec{n} - \vec{g}$. The greater the magnitude of $\vec{s}$, the more likely image space points in a narrow region in the vicinity of $\vec{n} + \vec{s}$ will be selected, while the smaller the magnitude of $\vec{s}$, the more likely image space points in the region widely scattered about $\vec{n} + \vec{s}$ will be selected. From published descriptions, it is not clear how successful steering is, but certainly it becomes more difficult as the dimension of image space — the number of parameters required to determine a point of the space — increases.

The dimension of image space also plays a critical role in a novel image generation technique introduced by Karl Sims [6]. For Sims, points or genotypes of image space are mathematical expressions, which are then *interpreted* to produce the visual phenotypes. Again small sets of phenotypes are selected, their genotypes are evolved, mated, and mutated by an evolutionary algorithm and then the results are displayed so that the image development cycle can continue. Since Sims' expressions are in fact rooted trees, image space is the infinite-dimensional space of rooted trees. Exploring this space is difficult. Even running on a supercomputer, the time to render a single genotype must be taken into consideration. Sims' version of an expert system prunes, or rejects entirely, rooted trees which predict prohibitive rendering times. The key to *coherent* search in this non-linear space is that many of the image processing functions stored at the nodes of the rooted trees representing the expressions are *neighborhood* image processing functions. Neighborhood image processing functions have the advantage of letting nearby pixels of the phenotype "communicate" with each other. Noise generators included among the image processing functions provide further global continuity to a rendered image. Finally, Sims' design posits phenotypes with intrinsic color, which means overall image design and image color are inextricably linked because the image processing functions "see" the pixels as RGB or HSV color

space vectors.

The "control problem" for searching images in these latter two systems is neatly codified by Margaret Boden, who formulates it in the context of agents [1]:

> [Sims] results are always "viable," in the sense that the newly transformed code will generate some visible image or other. But the process is utterly undisciplined. Although it could be used to help graphic designers come up with images they would never have thought of themselves, *it cannot be used to explore or refine an image space in a systematic way.* However, that is possible if the mutating agents are allowed to alter only the superficial parameters of the code. Significantly, these less powerful agents are preferred by [Latham] (italics added).

In the next section, we shall introduce a hybrid model which strives to make exploration of an image space of expressions a more disciplined process.

3. The Mathematical Model

To begin to develop a mathematical model for our expressions, we start with a set $\mathcal{V} = \{u, v, w, c, e\}$ of variables. These variables will reside at the *leaves* of our rooted trees. We will let (u, v, w) denote an arbitrary point in the unit cube, c denote a *constant*, and we reserve the variable e for "indirection" which will be explained in a later section. At *internal* nodes of our rooted tree, we store *pointwise* image processing functions in either one or two variables. Thus we have a set of image processing primitives, $\mathcal{F}$, defined on either the *unit* interval I, or on the *unit* square $I \times I$. Our formal definition of an expression adds a new ingredient which vastly enlarges image space while trying to establish a coherency between images rendered from nearby image points within the space. The idea is to attach "drift coefficients" to image processing primitives.

DEFINITION 3.1. An *expression* is defined recursively to be a *node* $(a\ b\ G)$, where $a, b \in I$, and either $G \in \mathcal{V}$ or $G \in \mathcal{F}$. If $G \in \mathcal{F}$, then each argument of G must again be a node.

EXAMPLE 3.2. Assume $\mathcal{F}$ contains the *normalized* sine and cosine functions, `nsin`: $I \longrightarrow I$ defined by $\texttt{nsin}(x) = 0.5{+}0.5{*}\sin(2\pi x)$ and `ncos`: $I \longrightarrow I$ defined by $\texttt{ncos}(x) = 0.5{+}0.5{*}\cos(2\pi x)$. Assume further that $\mathcal{F}$ contains the square root function `nsqt` and the function $\texttt{nmin}(x, y) = \min(x, y)$. Then, omitting superfluous parentheses, according to our definition a valid expression is:

```
0.05 0.38 nmin(0.54 0.09 nsin(0.42 0.52 u), 0.03 0.69
ncos(0.22 0.90 nsqt(0.04 0.63 v))).
```

DEFINITION 3.3. An expression $(a\ b\ G)$ is *evaluated* at a point $(x, y) \in I \times I$ using the recursive evaluation rule $\mathcal{E} : I \times I \longrightarrow I$ given by:

$$\mathcal{E}(a\ b\ G) = a * \mathcal{E}(G) + b \pmod 1,$$

where, if $G \in \mathcal{V}$, then

$$\mathcal{E}(G) = \begin{cases} 0 & \text{if } G = c \\ x & \text{if } G = u \\ y & \text{if } G = v \end{cases}$$

and, if $G \in \mathcal{F}$ is a function in d variables whose arguments are the nodes $n_1, \ldots, n_d$, then

$$\mathcal{E}(G) = G(\mathcal{E}(n_1), \ldots, \mathcal{E}(n_d)).$$

While the formal model explains how the imaging algorithm is implemented, its details may obscure the relatively straight forward computations that yield images. An example may help to clarify the algorithm.

EXAMPLE 3.4. To render the expression

```
0.05 0.38 nmin(0.54 0.09 nsin(0.92 0.42 u),  0.04 0.63 v)
```

as a 100×100 pixel image, we convert each pixel address (m, n) to the point with coordinates $x = m/100$ and $y = n/100$ in $I \times I$ and then evaluate the expression at (x, y). Therefore at the pixel with address, say $(85, 30)$, evaluation would proceed as follows:

$$\begin{aligned} E_1 &= \mathcal{E}(0.04\ 0.63\ \mathtt{v}) = 0.04 \cdot 0.30 + 0.63 = 0.012 \pmod 1, \\ E_2 &= \mathcal{E}(0.92\ 0.42\ \mathtt{u}) = 0.92 \cdot 0.85 + 0.42 = 0.202 \pmod 1, \\ E_3 &= \mathcal{E}(0.54\ 0.09\ \mathtt{nsin}(E_2)) = 0.54 \cdot 0.977 + 0.09 = 0.617 \pmod 1, \\ E &= \mathcal{E}(0.05\ 0.38\ \mathtt{nmin}(E_3, E_1)) = 0.05 \cdot 0.617 + 0.38 = 0.411 \pmod 1. \end{aligned}$$

Using a look-up table, the final value E is converted to a color which is then assigned to the pixel.

The previous example shows how the evaluation map $\mathcal{E}$ is used to manage the pointwise computation of an expression after inputs are assigned to those variables which are present at the leaf nodes of the expression. These inputs work their way up to the root to yield the result of the evaluation. This is the so-called "pandemonium model" where lower level processes shout their results to higher level processes. Figure 1 illustrates this by showing an example of a hierarchy of images implicitly formed at nodes during node evaluation. We must emphasize that evaluation is being performed over a discretization of the unit square. Notice how the presence of drift coefficients affects the node evaluation algorithm: After passing through the image processing function, but *before* being broadcast to the parent node, the intermediate result is perturbed by the linear equation determined using the node's drift coefficients, and then only the *fractional* part of that result is broadcast to the parent node.

4. Expressions as Genotypes

In preparation for exploring image space, an artist is initially presented with a randomly generated expression consisting of two or three nodes. This genotype is obtained through a "creation event." The artist then explores image space by following an evolutionary trail of this genotype by

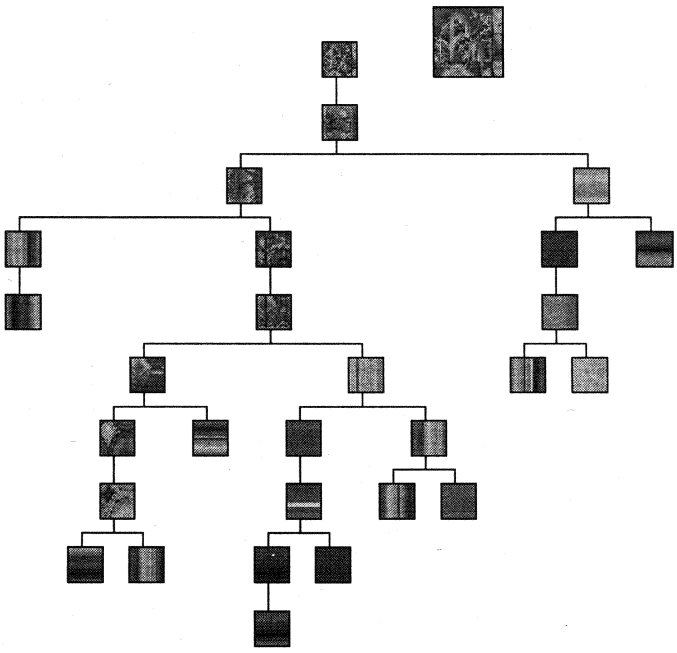

Figure 1: Image processing sequence showing images "bubbling up" to the root. (The final image at the foot is enlarged.)

tracking small populations of its descendants using evolutionary operators to manipulate genotypes while viewing phenotypes. More precisely, mating algorithms and mutation algorithms, together with special "catastrophe" algorithms are applied to one or two expressions chosen from the current population to produce new expressions which may, if so desired, be used to add new expressions to the population, or to provide replacements for existing expressions in the population. Some of these algorithms are biologically inspired, but others have been designed specifically to enhance image space exploration. Mating algorithms include various *crossover* algorithms for rooted trees borrowed from the field of Genetic Algorithms. They also include genetic exchange algorithms such as swapping, grafting, or cloning that model transcription and replication errors that are thought to occur during biological genotype reproduction. Mutation algorithms include "bit flipping" operators to randomly alter the contents of one or more nodes of an expression. Catastrophe algorithms provide for significant gain or loss of genetic material, thereby simulating developmental advance or retreat within the population on an evolutionary time scale. For the artist, however, the most powerful algorithms are the ones that subject genotypes to "genetic drift" by gently perturbing the drift coefficients at each node by small random amounts. Genetic drift allows the artist to slow the evolution of the genotype in the sense of Sims, yet explore "nearby" genotypes in the sense of Latham.

5. The Art of Evolving Expressions

In the previous section we noted that when evolving an image, the artist assumes control

following a creation event. The process leading to the development of a final image however is governed by artistic *expressivity*. Such expressivity might be guided by emotional response, subconscious motivation, or even simple curiosity. Before we examine some of the control techniques we have at our disposal for achieving this expressivity, it seems appropriate to include a slightly different point of view about the artistic nature of art-by-choice. This following excerpt is taken from an article by Tait [7] whose primary concern is with the broader question of whether purely computer-generated imagery is fine art:

> The work of Karl Sims stands out in bold relief against a background of technology posing as art... What then is the creative need fulfilled by Karl Sims algorithms?... I sense that he has opened a channel for exploration of beauty, a creative subject that has also become out of fashion in the high tech world. The exploration of beauty is however a timeless theme winding its way throughout human existence. It is a rare occasion when a new creative tool appears that allows us to reach out and touch beauty."

By definition, a phenotype is an evaluation of a genotype over a discretized version of the unit square. Therefore, the most important control technique takes place during the step where a visual image is obtained by transforming the numerical values resulting from the evaluation function into colors using look-up tables. This use of extrinsic color as opposed to intrinsic color serves to guide and hence to control image space exploration in two simultaneous but distinct ways, monitoring the image coloration in a preferred palette on one hand, while refining or developing the "structure" (i.e., overall design) of the image by switching among contrasting palettes on the other. We find color schemes with varying contrast properties are helpful, even necessary, for identifying emerging regions, shapes, contours, and their spatial relationships within phenotypes. In fact, contrasting palettes are of such critical importance, that organizing sequences of hues in non-traditional ways is our principal method for fully gauging image structure [3].

Familiarity with the mating and mutation algorithms provide additional controls. One example is revitalizing a large, stagnant genotype expression by "grafting" it onto a smaller genotype expression. There is a heuristic for why it is reasonable to expect this to be a successful strategy: We evolve the genotype for the most part by modifying its branches and leaves. Wholesale grafting develops its root structure. A second example is to employ a catastrophe algorithm to randomize the contents of a *portion* of the *leaves*. Exactly why this is successful is not entirely clear. Perhaps by keeping intact the core image processing functionality afforded by the genotype expression, a balance is struck between introducing new characteristics while retaining old ones.

When an evolved expression reaches a size of about thirty nodes, it begins to "stabilize" thereby setting the stage for a different set of artistic controls to come into play. These are controls for obtaining fine variations within the phenotype. For example, one technique is to evolve a *niche* population of expressions from one specimen. Each genotype in the niche population has a few nodes (either internal or leaf nodes) that have been altered by bit flippings from the original specimen. The technique proceeds by mating within this population using crossover operators. The design feature that enhances this fine-tuning control is the underlying *hidden* genetic drift algorithm that mildly perturbs all the drift coefficients after each and every mutation or mating operator is invoked. Figure 2 shows an example of a finished piece that was made using the techniques we have discussed.

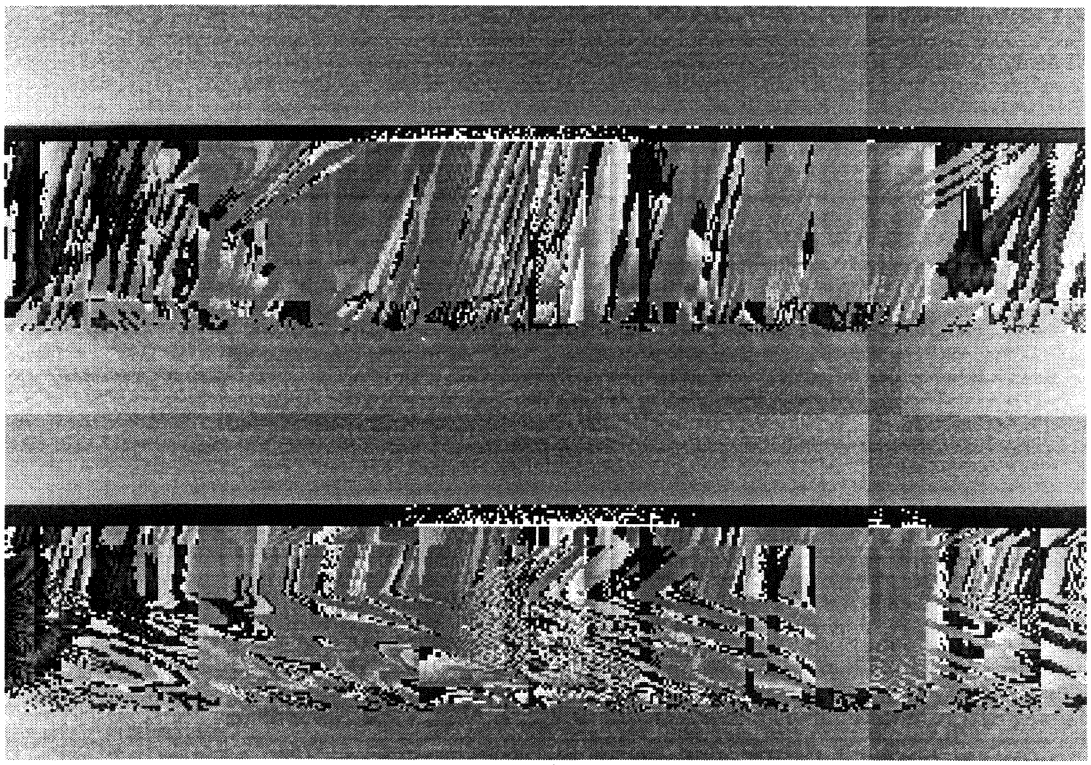

Figure 2: *Mixed Mode.*

6. Additional Features

We remarked previously that implicit in our overall design is the capability of rendering three dimensional images. We achieve this by simply using the variable w for the third spatial coordinate. After this is done, rendering proceeds by *voxelizing* the *surface* of the unit cube. Unfortunately, lighting and transparency problems must be dealt with at this juncture. Space considerations prevent us from covering the details behind surface-lit and volume-lit methods and their trade-offs.

Main stream computer art tools are strongly biased towards image compositing — constructing an image by building it up in layers. Our images are resolution independent. Resolution is governed by the discretization of the unit square alone, so our options in this area are somewhat limited. There is, however, a feature reminiscent of compositing that can be implemented. Expressions in which the variable e appears allow the artist to exercise the option of either treating this variable as a null input, or treating this variable as an input from a *background* expression, say H. Formally, at the point $(x, y) \in I \times I$,

$$\mathcal{E}(e) = \begin{cases} \mathcal{E}(H) & \text{if "compositing"} \\ 0 & \text{otherwise} \end{cases}$$

We think of occurrences of the variable e in an expression G as providing "sockets" for feeding the background image defined by H into G. While this has produced many successful images, we have no biological analogy to justify its inclusion within our framework. Hence we have no motivation or rationale for deciding in what situations it should be invoked. Further investigation into such

radical "extensions" of the basic technique is required.

References

[1] Margaret A. Boden, Agents and creativity, *Communications of the ACM*, **37** No. 7 (July, 1994), 117–121.

[2] Richard Dawkins, The evolution of evolvability, 201–220, *Artificial Life*, Christopher Langton (ed.), Addison Wesley, Reading, MA 1989.

[3] Gary Greenfield, An algorithmic palette tool, UR Technical Report TR-94-02, 1994.

[4] Lev Manovich, The engineering of vision and the aesthetics of computer art, *Computer Graphics*, **28** No. 4 (November, 1994) 259–263.

[5] Gregory J. E. Rawlins (ed), *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA 1991.

[6] Karl Sims, Artificial evolution for computer graphics, *Computer Graphics*, **25** (1991) 319–328.

[7] Will Tait, The space between: fine art and technology, *Computer Graphics*, **32** (Febuary, 1998), 17–19.

[8] Steven Todd & William Latham, *Evolutionary Art and Computers*, Academic Press, San Diego, CA 1992.

[9] Steven Todd & William Latham, Mutator, a subjective interface for evolution of computer sculpture, IBM UKSC Report 248, 1991.

[10] Mark Wilson, *Drawing with Computers*, Perigee Books, New York, NY 1985.