Evolving Line Drawings

Evolving Line Drawings

Ellie Baker
Margo Seltzer
Aiken Computation Laboratory[1]
Harvard University
33 Oxford Street
Cambridge, MA 02139

Abstract

This paper explores the application of interactive genetic algorithms to the creation of line drawings. We have built a system that starts with a collection of drawings that are either randomly generated or input by the user. The user selects one such drawing to mutate or two to mate, and a new generation of drawings is produced by randomly modifying or combining the selected drawing(s). This process of selection and procreation is repeated many times to evolve a drawing. A wide variety of complex sketches with highlighting and shading can be evolved from very simple drawings. This technique has enormous potential for augmenting and enhancing the power of traditional computer-aided drawing tools, and for expanding the repertoire of the computer-assisted artist.

1 Introduction

In his book, "The Blind Watchmaker," zoologist Richard Dawkins describes a computer program for evolving images of creatures he calls "biomorphs" [Daw87]. Each biomorph is produced from a compact genetic code that specifies the creature's particular characteristics. Contained in the code is the essential information necessary to create a bit-mapped image of the creature on the computer screen. Using a technique called *interactive evolution*, the computer and user collaborate to produce complex and varied insect-like creatures. With interactive evolution, the user selects from a series of biomorphs displayed on the screen the one deemed most aesthetically pleasing. By randomly mutating the genetic code of the selected biomorph, the computer creates a new generation of biomorphs to display. The new generation is again subjected to the aesthetic selection criteria of the user to produce the following generation. This cycle continues until the user "evolves" a pleasing creature.

Dawkins' idea has spawned several successful applications of interactive evolution to other image-design/construction problems ([Sim91], [CJ91], [TL92], [Sth91]). This paper presents an interactive evolution system for creating line drawings. The genetic representation and definition of the genetic operators is sub-

1. Email correspondence should be directed to ellie@das.harvard.edu

stantially different from previous interactive evolution systems, and it produces correspondingly unique results. The user interface is similar to other systems (especially Sims' [Sim91]and Dawkins' [Daw97]). An initial population of drawings, either generated randomly by the computer or input by the user, is displayed on the screen. From the displayed set the user selects one drawing for mutation or two drawings for mating. The mating and/or mutation operations are applied to the selected drawings to produce a new set of progeny drawings that supply the input for the next round of user selection. This process is repeated multiple times to "evolve" a drawing of interest to the user. Evolved drawings may be saved and later recalled for mating with other evolved drawings.

2 Interactive Evolution

Interactive evolution provides a powerful new technique for enabling human-computer collaboration. It is potentially applicable to a wide variety of search problems, provided the candidate solutions can be produced quickly by a computer and evaluated quickly and easily by a human. Since humans are often very good at processing and assessing pictures quickly, interactive evolution is particularly well suited to search problems whose candidate solutions can be represented visually.

Traditionally a genetic algorithm (GA) requires the specification of survival fitness criteria to be evaluated by the computer [Gol89]. This is typically one of the most difficult tasks in designing a GA. With interactive evolution the user performs this step, applying whatever complicated measure of fitness is desired. Unfortunately, including a human evaluator also severely weakens the GA because the human's speed and patience become new limiting factors, restricting the population size and possible number of generations. Despite this drawback, interactive evolution has been used to produce some astounding results that could not have been achieved easily by any other known method ([Sim91], [TL92]).

The beauty of interactive evolution is that the user does not have to state or even understand her own fitness criteria; she need only be able to apply them. This feature of interactive evolution is used very effectively in a system by Caldwell and Johnston for allowing a crime victim to produce a facial composite of a criminal suspect (CJ91). This system takes advantage of the remarkable human ability to recognize faces. A database of face parts (e.g., eyes, noses, mouths) is used to construct candidate faces, which are then rated by a human operator for their degree of likeness to the suspect. Based on these ratings, the faces are

recombined and mutated to produce a new generation of faces. This rating and procreation process is repeated until a likeness to the suspect is achieved. Caldwell and Johnston state that, while “humans have excellent facial recognition ability,” they “have great difficulty in recalling facial characteristics with sufficient detail to provide an accurate composite” [CJ91]. This is a perfect example of the ability to apply a complex fitness test without consciously understanding it. Since computers have historically performed poorly at face recognition (compared to humans), the system employs a particularly suitable division of labor between human and computer.

Sims’ system uses interactive evolution to create beautiful, abstract color images [Sim91]. Sims uses a parallel supercomputer to render the images at interactive speed. The genetic code for an image is a Lisp expression representing a sequence of image-processing functions (i.e., functions that take as input a set of pixel values and associated coordinates, and produce new pixel values as output). Sims uses a fixed set of image-processing primitives, and uses interactive evolution to evolve increasingly complex functions. The search space consists of all Lisp expressions that can be constructed from the primitives (but he filters evolved functions to exclude those whose rendering time would be too long). The mutation and mating operators restructure or modify the Lisp expressions and make random parameter changes. Sims also evolves plant forms by applying interactive evolution to L-Systems [LP89], grammars that describe biological models of plant growth [Sim91].

Other interactive evolution systems of note include: Dawkins’, which uses a recursive genetic structure that produces varied, but highly characteristic insect-like forms [Daw87]; Todd and Latham’s, which uses constructive solid geometry techniques to evolve “virtual sculptures” [TL92]; Smith’s, which uses a Fourier-series-based representation to produce bug-like curved line forms [Sth91]; and Oppenheimer’s, which produces life-like 3D tree forms using a recursive fractal representation similar to Dawkins’ [Opp89].

These systems use a variety of genetic representations to explore both infinite and large finite spaces. Each system relies on the ability to represent candidate solutions visually and on the human ability to evaluate these solutions quickly and in parallel. The evaluation criteria used are difficult-to-articulate personalized assessments of such poorly defined characteristics as “interesting,” “aesthetically beautiful,” “good likeness,” or “life-like”. These are terms whose definitions may vary drastically from person to person or even

change from moment to moment in the same person. It is this type of search problem for which interactive evolution provides an exciting new tool.

To date only a handful of interactive evolution applications have been built. These few applications have shown interactive evolution to be an interesting and useful tool, but there is still untapped potential in many other areas as well. One goal of this research is to add to current understanding of interactive evolution by applying it in a new domain. In doing so we hope to gain fresh insight into both its power and its limitations.

At the application level, the goal is to build a new kind of computer-aided drawing tool. Traditional tools use a compact, object-oriented drawing representation that makes it easy for a user to apply many operations to the drawing. Unlike a bit-mapped image, this high-level representation allows the user easy manipulation of individual drawing features, such as the ability to delete or modify individual lines, points, or other objects. However, creation of drawings in this format requires very good eye-hand coordination, and, for anything even slightly complex, a great deal of effort and tedium. Using these tools to create drawings beyond a certain level of complexity can be all but impossible, especially for a non-artist. The work presented here is intended to demonstrate the potential for using interactive evolution to augment and enhance the power of traditional computer-aided drawing tools and to expand the repertoire of the computer-assisted artist.

Section 3 describes the Drawing Evolver and shows how interactive evolution can be used to create complex drawings in a high-level format. Section 4 discusses our experience using the Drawing Evolver and Section 5 suggests areas for further investigation.

3 Drawing Evolver

The Drawing Evolver is an interactive evolution system written in the C programming language that runs under X Windows on a Unix workstation. Its basic components are: a structured representation for drawings; a means of producing an initial population of drawings; mating and mutation operators; and an operator that produces a drawing from its corresponding genetic code.

3.1 Drawing Representation

In the language of biologists, the genetic constitution of an organism is referred to as its *genotype*, and the physical appearance of the organism as its *phenotype*. In the Drawing Evolver, the organisms are drawings whose appearance (or phenotype) is determined by their genetic code (or genotype). The core of the system is the structure of the genotypes used to represent drawings. A genotype consists of an ordered set of "strokes," where each stroke is represented by an ordered set of points. A stroke may be loosely thought of as a mark made by a pencil without lifting it off the page. The stroke specification includes per-stroke parameters for such things as the method for connecting points (straight lines or spline curve) and symmetry type (horizontal, vertical, both, or neither). A set of symmetric marks are encoded as a single mark with a stated symmetry type. In this case several disconnected marks are still referred to as one stroke. The number of strokes in a drawing, as well as the number of points in a stroke, can vary, resulting in genotypes of varying length, and drawings of varying complexity.

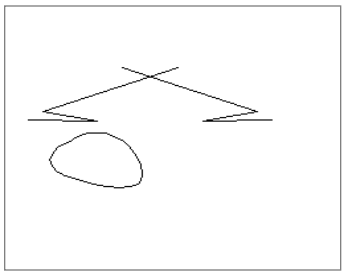
In theory, this representation provides for the possibility of any line drawing contained within the drawing frame. The number of possible phenotypes is very large, but finite. Since there are only two possible values (*on* or *off*) for every pixel in the drawing frame, and we use a 250 by 250 pixel square frame, there are a total of 2^{62500} distinct phenotypes (over 18000 orders of magnitude larger than Caldwell and Johnston's search space[2]). Figure 1 illustrates a simple drawing and its corresponding genotype. Table 1 specifies how drawing genotypes are interpreted.

To mutate a drawing, randomly chosen strokes or stroke points are moved, deleted, or added, and stroke parameters are randomly modified. When two drawings are mated, a randomly chosen subset of strokes from each of the parent drawings are combined to form a new drawing. These operations are described in more detail in Section 3.3 and Section 3.4.

3.2 Operation Modes

The system has two modes of operation that differ with respect to how an initial population of drawings is created. In *random mode*, the computer creates an initial population of randomly generated drawings. In *user-input mode*, the user specifies one or more input drawings to use for creating the initial population.

2. Caldwell and Johnston's space contains 34 billion possible composites [CJ91].



O H S 107 146 12 45
86 64 188 101 147 109 199 108
S N B 110 157 5 43
12 150 78 105 116 174

Figure 1: A simple example drawing (left) and its corresponding genotype (right). Although the drawing has three separate marks, it consists of only two strokes. The upper jagged stroke has the property of being "horizontally symmetric," so it consists of two separate marks that mirror each other across a vertical axis. Table 1 below explains how to interpret a drawing genotype**.**

INTERPRETING A DRAWING GENOTYPE

Each stroke is represented by two consecutive lines of text.

The *first text line* of each stroke gives the stroke parameters. These parameters are interpreted as described in the Stroke Parameter List below, and are ordered as shown in this list.

The *second text line* is an ordered set of the x, y coordinates of the stroke points (i.e., x1 y1 x2 y2 x3 y3...).

STROKE PARAMETER LIST

STROKE TYPE:
O = Open (first and last points are not connected)
S = Space Enclosing (first and last points are connected)
G = Glued To Next Stroke (last point is connected to first point of next stroke)

SYMMETRY TYPE:
N = No Symmetry
V = Vertical Symmetry
H = Horizontal Symmetry
A = Vertical and Horizontal Symmetry

POINT CONNECTION TYPE:
B = Spline Curve
S = Straight Lines

VERTICAL AXIS:
X-coordinate of the vertical reflection axis (to be used if the stroke is horizontally symmetric).

HORIZONTAL AXIS:
Y-coordinate of the horizontal reflection axis (to be used if the stroke is vertically symmetric).

PERTURBATION FACTOR (optional - if not given, a default is used)**:**
Maximum distance stroke or stroke points may be shifted during a single mutation.

MUTATION RATE (optional - if not given, a default is used)**:**
Probability that the stroke will be mutated during reproduction.

Table 1: This table describes how to interpret the ASCII text of a drawing genotype. The genotype is a "blueprint" that defines how to construct the drawing.

3.2.1 Starting With Randomly Generated Drawings

In *random mode*, the computer initially creates a "screenful" of random drawings (20 in the current version). The user can repeatedly request a new set of drawings until interesting ones are found. Random drawings may also be requested at any time during the evolution process, typically to be used for mating with an evolved drawing. Examples of three initial random drawings are shown in Figure 2. Figure 3 shows some drawings that were evolved using *random mode*. The butterfly forms were created with no

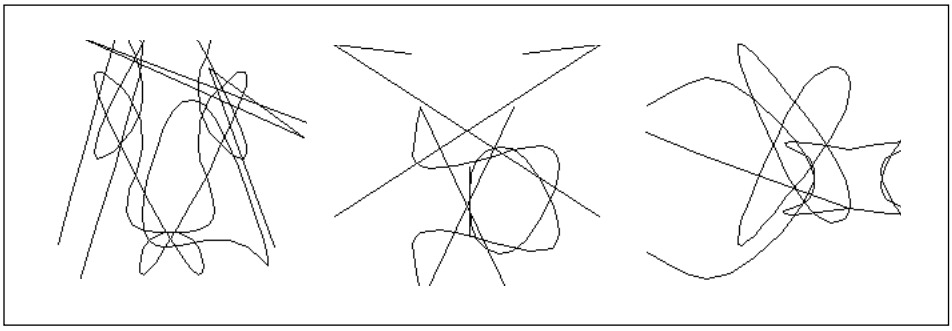


Figure 2: Randomly generated drawings. These three drawings were produced automatically, with random choices made for the number of strokes, the stroke points, and stroke parameters. A set of drawings like these may be used as an initial population.

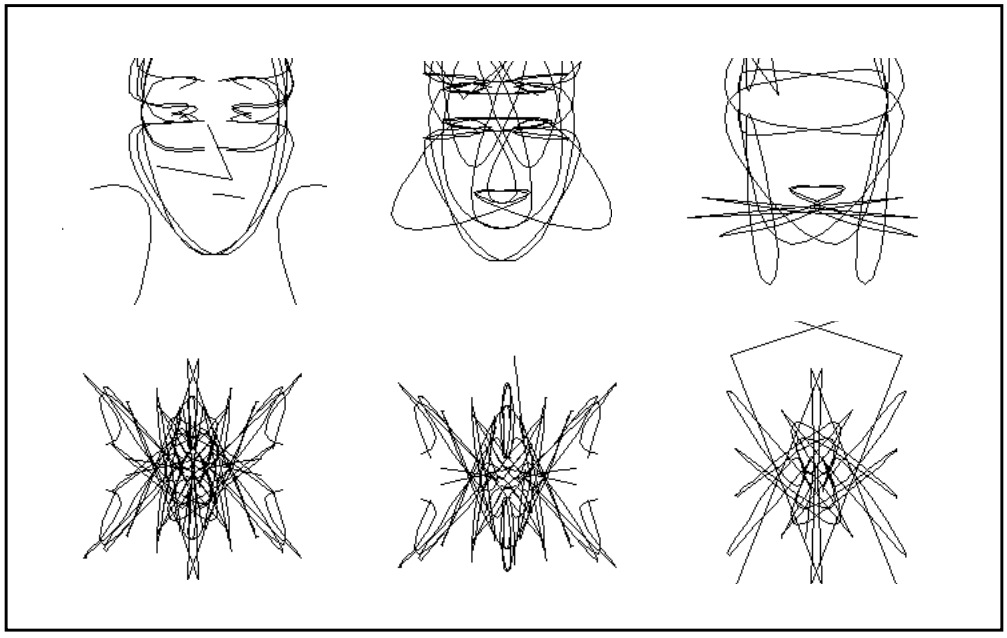


Figure 3: Drawings evolved from random drawings. The face drawings are related to each other (i.e., they were evolved during the same session), as are the butterfly forms.

preconceived goal in mind and took only a few minutes to produce. The face drawings were created with the specific goal of producing something face-like. This task proved to be much more difficult, taking over an hour to accomplish. However, once a face did emerge, it was quite easy to produce interesting variations on it. This experience motivated the addition of *user-input mode* described below.

3.2.2 Starting With a User-Input Drawing

In *user-input mode*, the user provides one or more drawings as a starting point. In this way, the system begins with a coarse solution (or a set of them), and the search is immediately focussed on a potentially rich area. If only one input drawing is given, it and a screenful of its mutated children make up the initial population. If none of the displayed children are sufficiently interesting, the user may reselect the initial drawing for mutation repeatedly to view new sets of mutated children.

The computed "average face" (from [Bre86]) shown at the top of Figure 4 was used in experiments as an initial input drawing and is the *sole original ancestor* of *all* drawings presented in the remainder of this paper.[3] An average face is a useful initial input drawing because it provides a central point for evolving a variety of different faces. Faces in general are an ideal subject matter for interactive evolution because of the specialized (but poorly understood) human ability to recognize and process them. The decision to limit examples to faces evolved from a single ancestor drawing was made to clarify the point that the variation achieved is a product of the interactive evolution process and not the result of starting from varied drawings. Figure 4 shows an assortment of face drawings evolved from the average face and illustrates the wide variation that can be achieved with the system. These drawings were produced from the average face by a simple sequence of mutations and matings. They were typically evolved in twenty to fifty generations, usually taking five to twenty minutes to create. Once a library of faces had been evolved, many interesting new faces were produced very quickly by combining them. The number of generations required is obviously very variable and depends heavily on the user's goals and intentions.

3. Brennan calculated average features from a large set of face drawings of real people, and constructed this "average face" for use in work on automating the creation of facial caricatures.

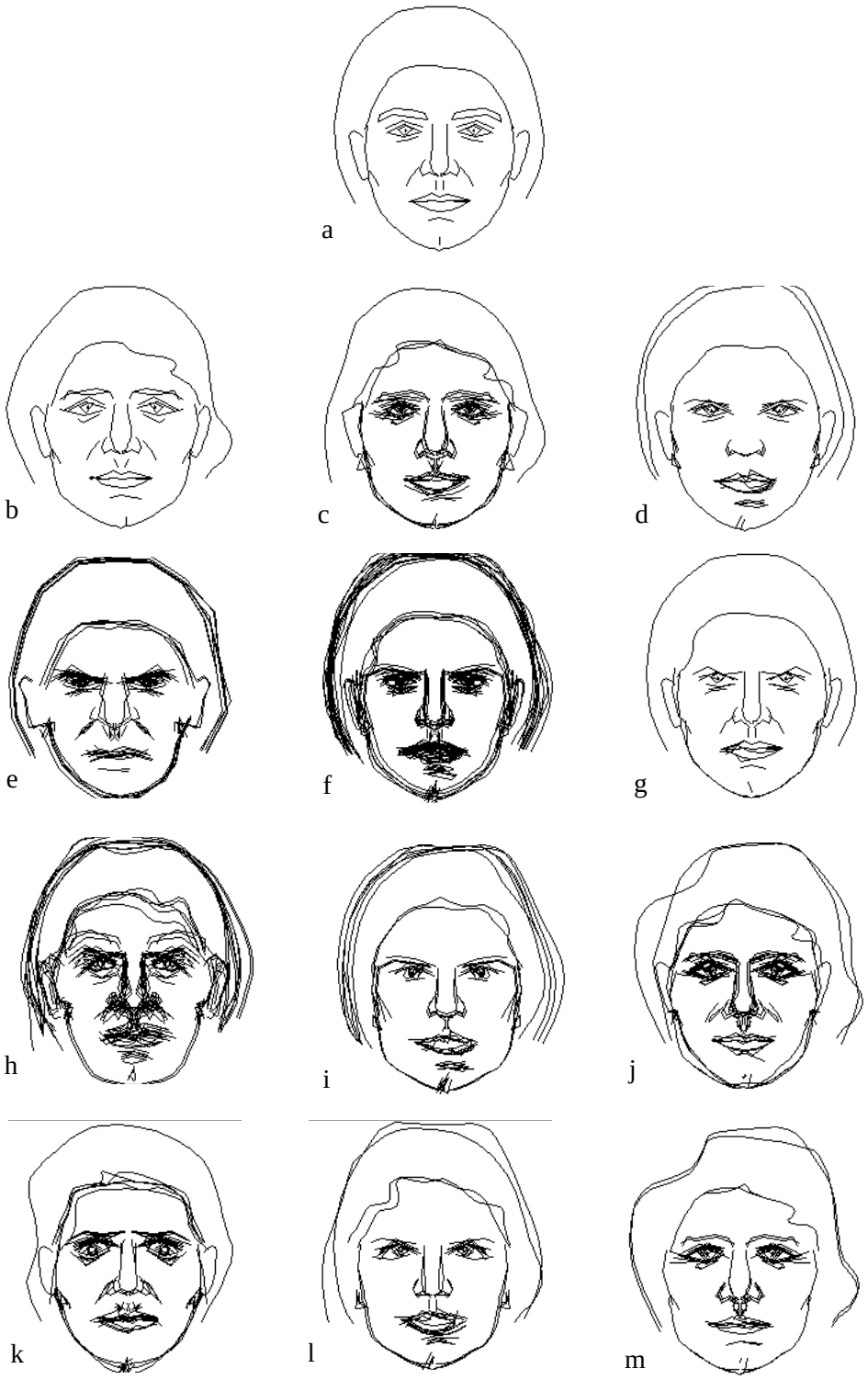


Figure 4: The average face (a) and twelve of its descendants (b-m). These drawings illustrate the wide variation possible in drawings evolved from a single ancestor. The drawings range in complexity from very simple (b, g) to bold charcoal-like sketches (f), to very intricate line drawings (h). Eyes can be lit with shiny highlights (j), some faces appear distinctly male (e, h) and others female (c, f, l, m). Some faces appear angry (e, g), pleasant (j), or horrified (k).

3.3 Mutation Operator

A mutated “child” drawing is produced from the genotype of its “parent” by randomly translating, adding, or deleting stroke points or entire strokes, and by randomly modifying stroke parameters (see Table 1). Random translations are effectively a means of jiggling around the strokes, much as an artist might do when trying out various small adjustments to the lines in a sketch. Modifying stroke parameters, on the other hand, generally causes more substantial structural changes to the drawing (for example, imagine changing the symmetry property of a stroke from vertical to horizontal). The probability of each type of mutation is individually controlled with adjustable system parameters. By setting the probability of a given mutation type to zero, it is possible to turn off that type of mutation altogether. When evolving drawings in *random mode*, all mutation types were generally used; in *user-input mode*, stroke parameter mutations were turned off. With the average face as an input drawing, it was not generally useful to change the basic properties of the original face (for example, since one normally always wants two eyes, two ears, etc., it didn’t make sense to allow changes to the symmetry properties of the face). By limiting mutations to those that jiggled around the existing lines without changing basic properties, it was possible to retain the ‘face-ness’ of the original drawing while still getting a great deal of interesting variation.

The genotype specifies a *perturbation factor* for each stroke, which restricts the distance (in pixels) that the stroke or stroke point may be moved during a single mutation. The genotype also specifies a mutation rate indicating the probability that a given stroke will be mutated during reproduction. In *user-input mode* the perturbation factors and mutation rates for the input drawing may be tailored to the specific subject matter. For example, the hair outline of the average face was given a larger perturbation factor than the eyes and other small features. If the small features are given too large a perturbation factor, they can jump off the face in one mutation. If the hair outline is given too small a perturbation factor, it’s difficult to get any significant variation from the bathing-cap look of the original drawing. The mutation rate for the hair was also set higher than for other parts of the drawing. The perturbation factors and mutation rates can also be mutated, but we have not yet experimented with this capability[4].

Figure 5 shows an example set of single step mutations starting with the average face. This is how the initial screen might look when the average face is given as the input drawing. The only active mutation types

4. This might be useful for automatic tailoring of perturbation factors and mutation rates.

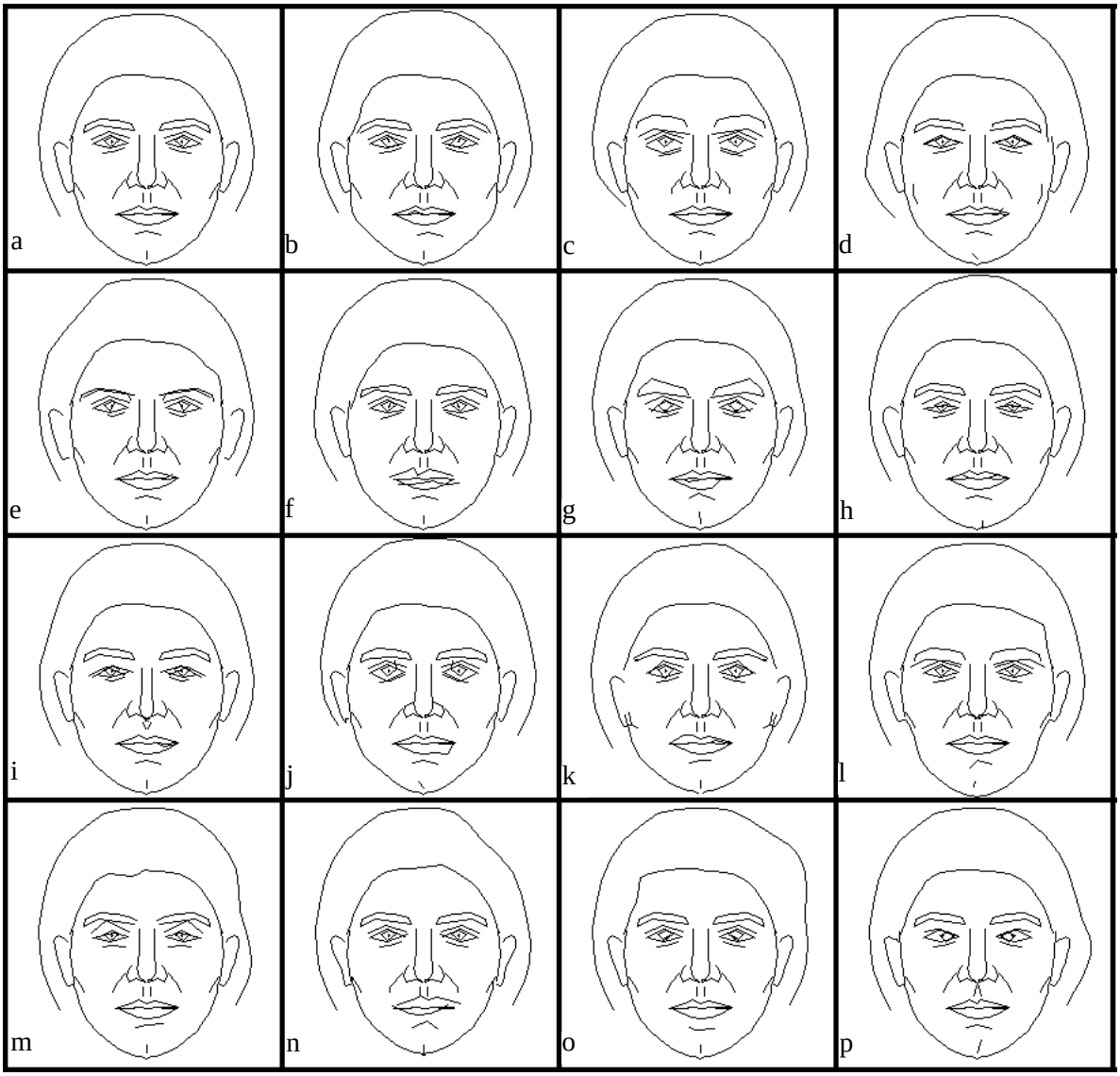


Figure 5: Example of single step mutations starting with the average face. The original average face is shown in (a). This figure is a segment of the screen from an actual run of the system in which the average face was given as an input drawing. Note that small changes in just a few strokes of a drawing can subtly change the appearance of the face. For example, the eyelid mutations in (m) give it a sleepy expression, the point translations in the jaw line of (b) make the jaw more angular and masculine, and the eyebrow mutations in (g) create bushy eyebrows.

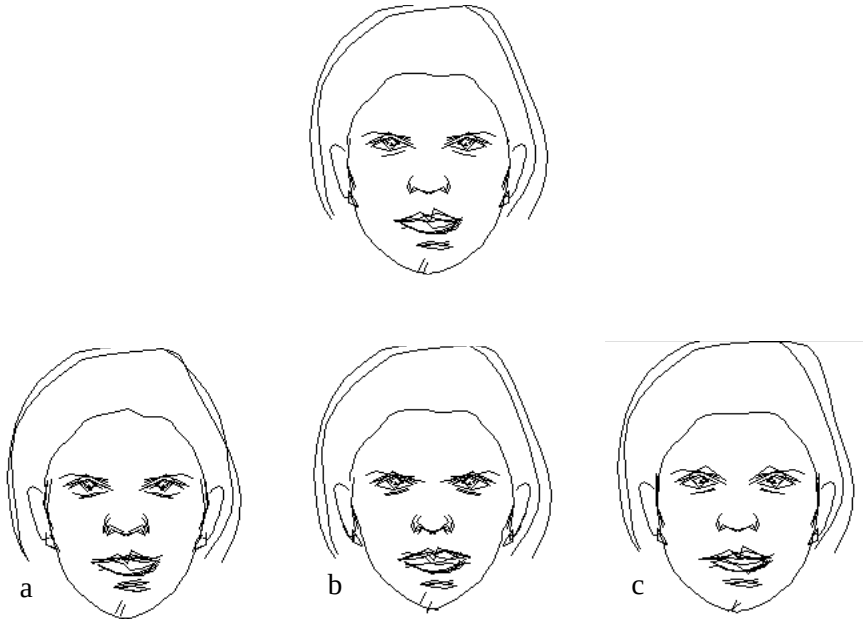


Figure 6: Mutations from an evolved face (parent above, children below). These particular mutations cause very subtle differences in facial expression, especially around the eyes. The mutations in (b) create a somewhat angrier look than the parent, whereas (a) has a softer friendlier appearance. The eyebrow mutations in (c) create a slightly perplexed look.

in this example are stroke deletions and translations, and point additions, deletions, and translations, with translation more likely than other mutation types. Figure 6 illustrates the kinds of subtle mood and expression changes that the mutation operator can produce in a more complex, evolved face.

3.4 Mating Operators

A mating operator takes two parent drawings as input and uses them to produce a child drawing. The basic approach is to choose a subset of the strokes from each parent and combine them to form the child. Although we experimented with a number of different mating operators, only two are used in the current system. Both schemes use a kind of uniform crossover [Sys89], particularly applicable here because the order of strokes is not necessarily important.[5] The primary difference between the two schemes is that, in one (referred to as *variable-length mating*), it is assumed that the child may have any number of strokes (either more or fewer than its parents, or the same number). In the other (referred to as *fixed-length mat-*

5. Unless some of the strokes have the property of being connected to the next stroke (and in the face examples, none do), reordering strokes will not produce any change in the drawing. Note that this is not true of stroke points, where the order always specifies a connecting sequence.

ing), it is assumed both parents have the same number of strokes, and that the child must also have this number of strokes.

3.4.1 Variable-Length Mating

In general, drawings can have any number of strokes, and their order is unimportant. Hence, combining *any* two subsets of the strokes from each of two parents would produce a legitimate child. To mate two drawings in this setting, we independently consider each stroke of each parent for inclusion in the child. If a fair coin is used to make the inclusion decision, this process will typically produce a child with approximately half of the strokes from each parent. Since this is not necessarily the desired outcome, we randomly choose a weight (or bias) for the coin, with a different weight chosen for each parent. If the weight is chosen to be any value between 0 and 1, the number of strokes in the child might be as many as the sum of the number of strokes in the parents, or as few as one.[6] Hence, a child may look much more or much less complicated than its parents. In practice, we chose to limit the weights to values between .3 and .7, so that a drawing can differ in complexity from its parents, but not too drastically. Depending on the coin weights chosen, the child may have more in common with one parent than the other. Figure 7 and Figure 8 illustrate variable-length mating.

3.4.2 Fixed-Length Mating

With *fixed-length mating*, parent drawings have exactly the same number of strokes, and the *nth* stroke in the mother corresponds to the *nth* stroke in the father. This condition must be true in the initial population, and it is maintained thereafter. Thus the number of strokes remains constant from generation to generation. To mate two drawings, one of each two corresponding strokes is included in the child. Each choice is made randomly with a fair coin.[7] Figure 9 illustrates fixed-length mating.

Fixed-length mating was developed when the average face was introduced as an input drawing. Initially, it seemed that the child should contain a nose or eye stroke (for example) from one parent or the other, but not both. This turned out to be useful in some situations, but not strictly necessary (as demonstrated in Fig-

6. We wisely insist that a drawing must have at least one stroke.
7. It might also have been interesting to randomly weight the coin, as was done with variable-length mating.

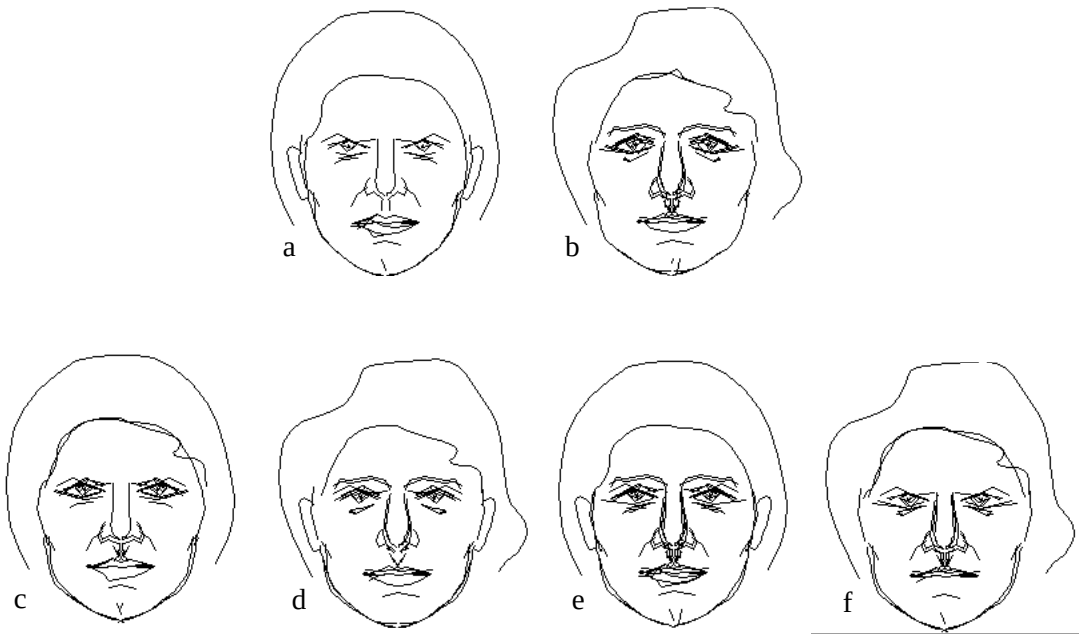


Figure 7: Variable-length mating (parents above, children below). Note that the children can inherit components of individual features or expression from both parents. In this case, parent (a) has an angry expression and looks more male than female, while parent (b) looks female. Child (c) inherited some of the angry expression from (a), but the darkly outlined eyes and curly lower hairline from (b), making it look more feminine. Assessments of this kind are clearly subjective.

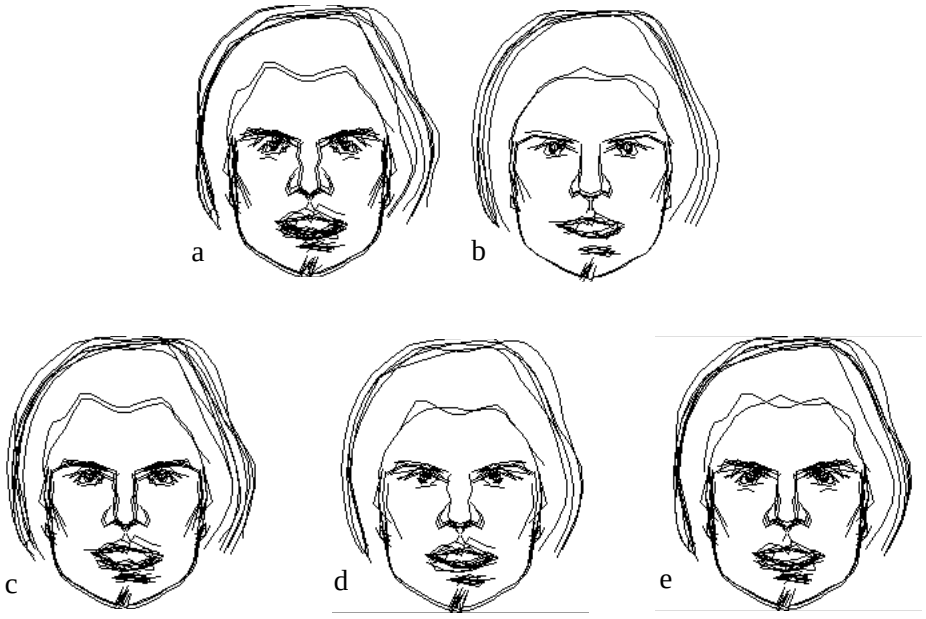


Figure 8: Variable-length mating (parents above, children below). These two parents are close relatives. The user evolved parent (b) to look female and evolved parent (a) from (b) to look male. They were then mated together in an attempt to produce an androgynous face. The user felt that child (c) was the most successful androgynous face. Note that the eyes in (c) and (d) combine features from both parents, and that the noses in all three children are subtly different from each other and from the parents.

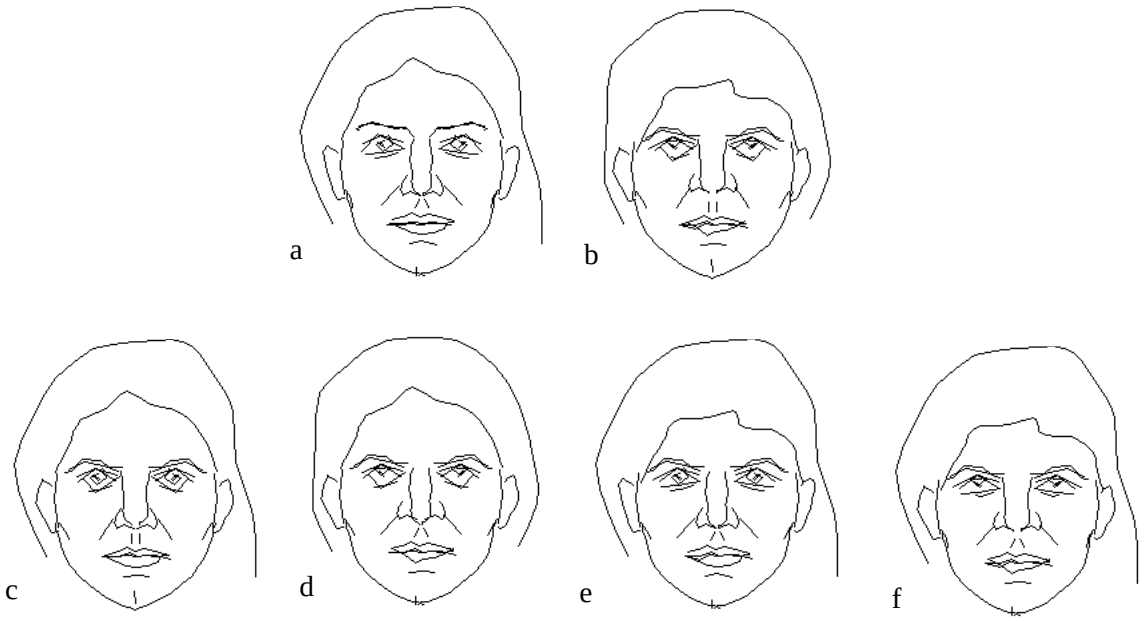


Figure 9: Fixed-length mating (parents above, children below). In contrast to variable-length mating, these children receive (for example) the hair outline stroke from parent (a) or from parent (b), but never from both. Child (f) has inherited the hair outline from parent (a), but the lower hairline from parent (b), creating a new hairdo.

ure 7 and Figure 8). The relative advantages and disadvantages of fixed and variable-length mating are discussed in Section 4.

4 Discussion

There are many factors that influence the success of an interactive evolution system. One measure of success is whether the system is fun and engaging to use. Another measure is whether or not it produces interesting or useful results. Yet another is how effectively the user is able to control the direction of the evolution toward a specific goal. Interactive evolution relies heavily on the user, so each person's experience is different. This section presents some observations on what it feels like to use the Drawing Evolver and an assessment of factors that contribute to its successes and failures.

Since the user spends a lot of time evaluating drawings, providing a reasonably varied population at each generation is an important aspect of engaging a user. If differences between drawings are too small and subtle, it is difficult to distinguish between them and make choices. On the other hand, if differences are too numerous and substantial, it is impossible to maintain the successful parts of a drawing from one gen-

eration to the next. The mutation rates and mating/mutation operators determine the amount of variation present. Overly high mutation rates can cause drawings to “jump out of” the user’s space-of-interest in one generation. For example, a face drawing will lose its “face-ness” very quickly if the facial features can move around too much. The per-stroke mutation rates and perturbation factors were useful additions to the genotype because they improved the system’s ability to evolve varied faces without requiring the system to understand anything about what a face is.

Another factor influencing the quality of the candidate solutions presented to the user is the appropriateness of the representation for the particular search space. The representation used in the Drawing Evolver is very general (i.e., it can represent any drawing at all) compared to Caldwell and Johnston’s (whose genotype represents faces only) [CJ91]. But if we wish to limit our search to faces, the user must filter out the drawings that don’t look face-like, while Caldwell and Johnston’s users always see only valid faces. Of course, having a larger search space can also be an advantage; it provides the potential to evolve facial expressions and cartoon or animal faces, as well as many other things. Thus the choice of representation is likely to involve a trade-off between narrowing the search space and limiting the possible solutions.

Another difference between the Drawing Evolver’s representation and Caldwell and Johnston’s is that, in the Drawing Evolver, a face may be represented at a finer level of granularity. For example, if a nose is composed of several strokes, it is possible to combine parts of one nose with parts of another. A finer level of granularity makes for interesting (and somewhat life-like!) facial recombination effects.

The mating and mutation operators also have an impact on the variation and quality of drawings. While experimenting with different mating operators, it was surprising to discover that variable-length mating worked well applied to mutations of the average face. Interesting effects, such as shading and highlighting, began to emerge in the evolved drawings, and the level of intricacy of a drawing could increase. If a face acquired two corresponding nose (or hair, eyelid, etc.) strokes, this often had the effect of producing a sketchy version of the face, with a more three-dimensional and textural feel. These are effects that an artist can work very hard to achieve in a drawing, and seeing them emerge in drawings produced by a computer is very exciting. A drawback to variable-length mating is that evolved drawings can become more and more complex and eventually become too delicate for further mutation. The results may be something like what artists refer to as “overworking” a drawing. In the current system no effort is made to eliminate dupli-

cate strokes, which may contribute to this problem. It might be helpful to have mutation rates that automatically decrease as the number of strokes in a drawing increases. Unfortunately, fixed-length mating, which eliminates the overworking problem, is less useful for producing highlighting, shading, and textural effects, and it limits the drawing's level of intricacy. As a compromise solution, many of the face drawings shown in this paper were evolved using fixed-length mating in the early stages and variable-length mating in the later stages.

In the Drawing Evolver, as in Sims' system [Sim91], each new generation is a collection of offspring from just one or two parents. This is in contrast to traditional GAs where the more fit solutions are statistically more likely to reproduce, but less fit solutions also reproduce, and the new population is the result of matings between many different parents [Gol89]. Typically, mutation rates are very low. The traditional approach helps maintain variation in the population and helps eliminate premature convergence. The assumption is that the population is large, and that most of the components necessary to produce an optimal solution are already present in the initial population. These assumptions do not apply in our setting, since the initial population is quite small, and, when starting with an input drawing, already quite uniform. Also, there is generally no single optimal solution, but rather many acceptable outcomes, and the system is best used as a tool for exploring this very rich space. In this situation, mutation is an extremely important avenue for finding new solutions, so outrageously high mutation rates (e.g., 25%) were used compared to those used in traditional GAs (and in biological evolution). With the Drawing Evolver, the user has partial responsibility for maintaining variation in the population. A typical approach to evolving a face drawing is to evolve one face with some interesting characteristics, save it away, and then start again to evolve something quite different. Once a small library of evolved faces have been created, they can be mated together to produce quickly many new and varied faces.

The system is generally easier and more pleasant to use for browsing the search space than it is for evolving a specified goal image. This is no surprise since the space contains a myriad of interesting drawings, but a comparatively small number that meet some narrowly specified set of requirements. The narrower the goals, the tougher the search problem. Caldwell and Johnston report success at using their system to evolve a likeness to a criminal suspect, but they use a very tightly constrained search space, and a genotype designed with this specific purpose in mind. The Drawing Evolver, with its much bigger and less constrained search space, is not particularly good at this task. Attempts to start with the average face and

evolve a reasonable likeness to a particular person were not very successful. On the other hand, more loosely specified goals, such as a plan for gender and facial expression were fairly easy to carry out.

5 Future Work

A challenge for interactive evolution systems is to find techniques that reduce the amount of work required of the user. One approach is to automatically filter evolved solutions and display only those that are likely to be of interest. Sims, for example, filters evolved images to eliminate those that would take too long to render [Sim91]. Other kinds of filters might make it possible to confine a search to a particular area. For example, ideally one would like to be able to produce a filter that confined a Drawing Evolver search to "face space." One possible approach to this problem is to add evolvable constraints to the genotype, and to use interactive evolution to find the right settings for them. The drawing evolver genotype includes evolvable perturbation factors and mutation rates that constrain the changes that can occur in a single mutation step. It might also be possible to impose constraints on the dimensions and locations of individual drawing strokes, or on their relationship to other strokes. If the constraint parameters are themselves evolvable, a user could select for constraint settings that did the best job of confining drawings to "face space." The evolved constraints would be a kind of pattern recognizer (in this case, one for recognizing the face-pattern). Since people are much better at applying complicated pattern-recognition criteria than they are at articulating them, interactive evolution may provide a useful method for allowing a user to specify such criteria simply by applying them.

6 Conclusions

The Drawing Evolver allows a user to produce a wide variety of complex sketches with highlighting and shading from a single very simple ancestor drawing. These drawings would be quite difficult and tedious (if not impossible) to produce with most conventional object-oriented computer-aided drawing tools. The user, in a reactive rather than proactive role, is responsible only for selecting among sets of drawings produced by the computer. The user need not have any drawing skill or eye-hand coordination (at least no more than is necessary for selecting drawings with the mouse), but good observational and visual skills are useful to be able to distinguish between sometimes subtly different drawings. The user forfeits absolute control over the outcome, but gains an extended repertoire of possible results.

Interactive evolution is an important new tool for using the computer as a creative collaborator in the exploration of large search spaces. While empowered by human-evaluated fitness testing, it is also limited by the slow pace of interactive use. Despite its limitations, interactive evolution has again proved a powerful tool in a new setting, adding more evidence to suggest that its full potential is yet to be explored.

7 Acknowledgements

Thanks to Jim Clark, Ted Nesson, Joe Marks, Karl Sims, Steve Smith, and Peter Todd for helpful discussions and encouragement. And special thanks to Karl Sims, whose work inspired this research.

8 References

[Bre86] S. Brennan. Cited in Computer Recreations by A. K. Dewdney. *Scientific American,* Vol. 225, October 1986.

[CJ91] C. Caldwell and V. S. Johnston. Tracking a Criminal Suspect Through Face Space With a Genetic Algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 416-421, 1991. Morgan Kaufmann Publishers.

[Daw87] R. Dawkins. *The Blind Watchmaker.* W.W. Norton and Company, New York, London, 1987.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison-Wesley, 1989.

[LP89] A. Lindenmeyer and P. Prusinkiewicz. Developmental Models of Multicellular Organisms: A Computer Graphics Perspective. In *Artificial Life,* edited by C. G. Langton, Proc. Vol. VI, pages 221-250. Addison-Wesley, 1989.

[Opp89] P. Oppenheimer, The Artificial Menagerie. In *Artificial Life*, edited by C. G. Langton, Proc. Vol. VI, pages 221-250. Addison-Wesley, 1989.

[Sim91] K. Sims. Artificial Evolution for Computer Graphics. *Computer Graphics*, 25(4):319-328, July 1991.

[Sth91] J. R. Smith. Designing Biomorphs with an Interactive Genetic Algorithm, *Proceedings of the Fourth International Conference on Genetic Algorithms,* pages 535-538, 1991. Morgan Kaufmann Publishers.

[Sys89] G. Syswerda. Uniform Crossover in Genetic Algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms,* 1989. Morgan Kaufmann Publishers.

[TL92] S. Todd and W. Latham. *Evolutionary Art and Computers.* Academic Press: Harcourt, Brace, Jovanovich, 1992.