

Chapter 6

MAKING INTERACTIVE EVOLUTIONARY GRAPHIC DESIGN PRACTICAL

A Case Study of Evolving Tiles

Carl Anderson¹, Daphna Buchsbaum¹, Jeff Potter¹, and Eric Bonabeau¹

¹*Icosystem Corporation, 10 Fawcett Street, Cambridge, MA 02138, USA*

Abstract This chapter describes interactive evolutionary design, a powerful technique where one marries the exploratory capabilities of evolutionary computation with the esthetic skills and sensibility of the human as selective agent. Interactive evolutionary design has the potential to be an enormously useful tool to graphic designers. However, in order for it to become commonly used, a number of barriers to use must be overcome, including making it both simpler to understand (e.g., more intuitive genotype-phenotype mappings) and with greater user-control (e.g., allow the user to lock “perfect” elements of a design). In this study we explore these ideas—how can one make interactive evolutionary design more appealing, useful and practical?—by developing a design tool “evoDesign” that employs a genetic algorithm to evolve designs. As an illustrative test case, evoDesign is used to evolve tiles used for walls or floors or as a repeating unit for fabrics and wallpapers.

Keywords: Interactive evolution; graphic design; tiles; evolutionary art.

1. INTRODUCTION

Evolutionary art, with human esthetic choice as the driving selective agent, has been around for many years and has been used to produce some stunning visual images (McCormack 1993; Rooke 2005; Sims 1991; Todd and Latham 1992; Whitelaw 1999, 2004). However, these images, which are often very organic and fractal-laden, are not necessarily appropriate for many practical applications; for example, they may not be something one would want to tile a bathroom with. In most cases, these artistic explorations are conducted by professional computer scientists or artists, people who are striving to create “foreground art” and who need not be concerned by mass appeal and salability, nor with the usability of their evolutionary tool.

However, interactive evolutionary design (Takagi 1998, 2001) has the potential to be an enormously useful tool to graphic designers and individuals designing clothing and other goods for mass markets, *if* it can be made more practical and palatable to those practitioners, many of whom do not have a quantitative background. In this study, we explore some of the issues of making 2-D graphic design through interactive evolution more practical and appealing in the real world, focusing on a tool for evolving floor, wall and fabric tiles as an illustrative test case.

In order for interactive evolution to become a commonly used graphic design tool, a number of barriers-to-use presented by existing systems must be overcome. One of the critical problems of work to date, at least in terms of interactive evolution as a *practical* tool for graphic designers, is that most evolutionary art is based upon genetic programming. This can be a very hard concept to grasp and to explain to non-expert end-users; for many people, genetic programming is very mathematical and very abstract. A genetic algorithm (GA) on the other hand is far simpler and *encapsulates* the same set of actions that a graphic designer performs when creating and modifying a design: adjusting the position or rotation of an element, altering the hue, transparency or color of a component and so on. For this reason, we use a simple GA rather than genetic programming.

Another problem with work to date is that it is often hard to control individual elements of a graphic independently. When using genetic programming, the picture is usually generated from a single equation (e.g. a LISP expression) in which all the elements are intertwined; it is often hard to fix one esthetically pleasing element while allowing other elements to evolve (this is also true of other multi-equation systems such as L-systems)—in short, genetic programming is brittle. This has two important consequences. First, a graphic designer may feel a loss of control. The goal of interactive evolutionary design is to marry the exploratory skills of evolutionary computation with the esthetic skills of the human as selective agent. A user who sees a pleasing element but who is unable to retain it in the design is likely to get frustrated with the system very quickly. Second, and related, the designer may feel threatened by this technology as a tool that is taking away their job. The goal should be to develop a tool that is complementary, perhaps supplementary, to the designer. The designer should be in control, either using the tool to generate some initial ideas or us-

ing interactive evolution to explore the design space around an already existing idea. The designer must be the most powerful driver of the system through the design space.

In this study, we attempt to put ourselves in the mind of a skilled graphic designer, one interested in exploring a particular design space, perhaps using the tool to generate initial ideas within some user-defined boundaries. Alternatively, they might be starting with a pre-existing idea, some graphical “digital amber” (Rooke 2005; Whitelaw 2004), which they would like to perfect. They want to be able to lock “perfect” elements as they evolve (Takagi 1995), or even edit them directly. And, they would like to understand the basic principles of the system. In other words, we develop a system that is intuitive, useful and practical, and (in hindsight) adopt a number of the “ten steps to make a perfect creative evolutionary design system” (Bentley and O’Reilly 2001) resulting in a system that happens to be much simpler to understand and intuitive to use.

2. GENETIC ALGORITHM

2.1 Phenotype

In this design tool, a phenotype consists of a panel, a “tile” upon which is drawn a set of simple graphical elements: colored horizontal and vertical stripes, circles, squares and rectangles. Four such tiles are shown in Figure 6-1. Each tile is toroidal or wrapped so that parts of elements that overlap one margin appear at the opposite margin. This guarantees that all designs that evolve constitute a repeatable unit (i.e., are tile-able in the strict mathematical sense) that is suitable for fabrics, wallpapers and so on (but see later discussion). The *evoDesign* tool contains a feature that allows one to see a chosen panel tiled across the whole screen at a user-controlled density (see Figure 6-2).

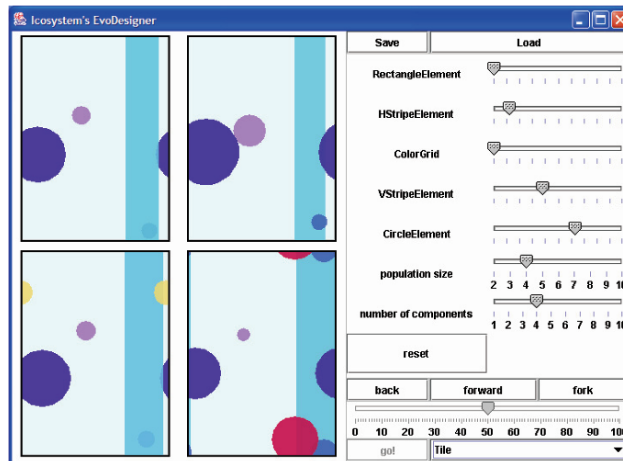


Figure 6-1. The evoDesign user interface. The four panels on the left constitute a population of tiles being evolved. On the right, sliders determine the relative proportion of the different graphic element types, the next two sliders, the number of panels and the number of graphic elements within each panel. “Back” and “forward” allow the user to revisit previous generations.

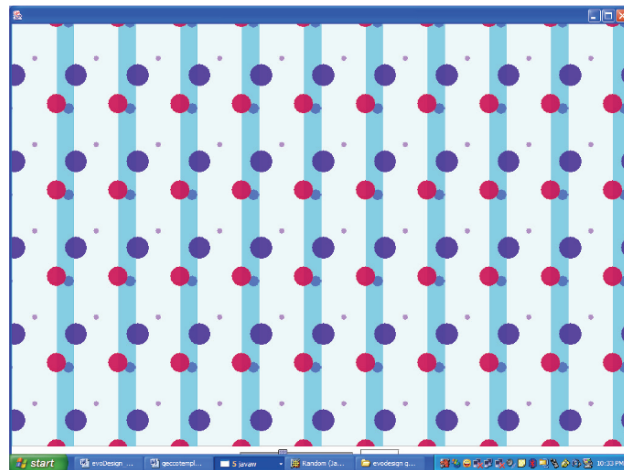


Figure 6-2. A tiling of the bottom right tile of figure 6-1.

2.2 Genotype

A genotype i , G_i , consists of a vector of graphic elements (“chromosomes”)—an individual circle, square etc.—each of which has a number of evolvable attributes (“genes”). Every graphic element has an evolvable four-gene color consisting of a red, green, blue (RGB) and α -(transparency) value, all in the range of $\{0, 255\}$. The evoDesign tool is used to evolve the number, size, shape, location, color and transparency of the graphical elements within a set of tiles.

2.2.1 Circle. A circle contains three other evolvable attributes: an (x, y) coordinate for the top left of the circle’s bounding box—the panels are normalized so that all x and y values are $(0, 1)$ —and a bounded diameter $d \in (0.05, 0.5)$. In short, a circle’s genotype is the vector $(R, G, B, \alpha, x, y, d)$.

2.2.2 Rectangles and Stripes. A rectangle contains four other evolvable attributes: an (x, y) coordinate for the rectangle’s upper-left corner, a width (w) and a height (h), both with a lower bound of 0.05. In short, a rectangle’s genotype is the vector $(R, G, B, \alpha, x, y, w, h)$.

A vertical stripe is simply a special case of a rectangle where $y = 0$ and $h = 1$, both of which are non-evolvable. Similarly, a horizontal stripe is a special case of a rectangle where $x = 0$ and $w = 1$, again, both are non-evolvable. The maximum width of a stripe is 0.3.

2.2.3 Color Grid. A color grid consists of a regular grid, usually 4×4 , of colored rectangles. The color grid’s only evolvable attributes are the color and transparency of its component rectangles.

The color, size and position of all graphical elements (except those of a color grid) in a tile may be “frozen,” i.e. made non-evolvable, from the user interface.

A particular gene may be frozen per element (e.g., the color of one circle) or throughout the entire tile (e.g., all elements have their color frozen, while size and position continue to evolve).

2.2.4 Drawing. The elements of G_i are drawn consecutively on tile i with later elements potentially overlapping or obscuring earlier elements, where the color grid constitutes one element. Hence, a final implicit attribute of a graphical element within G_i is its position or index which corresponds to a graphical “level.”

Except for frozen (x, y) coordinates, all attributes of graphical elements are independent. Hence, initialization of a tile simply consists of generating a given number of graphical elements, each with a random (bounded) value.

2.3 Mutation Operator

A genotype is mutated by randomizing the order of its graphic elements and then mutating each element. In addition, with a certain probability (here, $1/3$) a randomly chosen element is removed from the genotype and with a certain probability (here, $1/3$) a new graphic element is created and added to the genotype.

Each non-frozen gene that is mutated is done so uniformly across some half-range r around its current value and then bounded as necessary. That is, a gene g with current value $g(t)$ and bounds g_{low} and g_{high} takes on a new value,

$$g(t + 1) = \max(g_{low}, \min(U(g - r, g + r), g_{high})). \quad (1)$$

The following default values are used: $r_x = r_y = 0.05$; $r_R = r_G = r_B = 5$; $r_\alpha = 60$, $r_d = 0.1$, and $r_w = r_h = 0.05$.

2.4 Crossover Operator

In addition to mutation, we also use a crossover operator (Bentley 1999). Suppose that one parent genotype G_i contains n_i graphic elements while the other parent genotype G_j contains n_j elements and that $n_i \leq n_j$. Their offspring

G_k will have $n = U\{n_i, n_j\}$ genes, i.e. a length between that (inclusive of the bounds) of the two parents. The first $U(0, n_i)$ of which are randomly chosen (without replacement) from G_i with the remainder randomly chosen from G_j . In brief, an offspring is created of similar length to its parents, with some random proportion of its components coming from one parent and the remainder from the other. Figure 6-3 gives an example of the crossover operation.

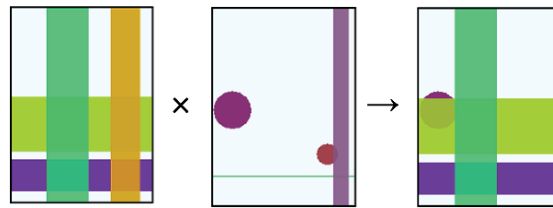


Figure 6-3. Crossover between two parents.

3. INTERACTIVE EVOLUTION

The evoDesign tool contains a population of 2 to 10 genotype tiles (Figure 6-1 contains a population of size 4). The user can evolve the population in a variety of ways. First, she can use mutation only. In this case, the user selects a tile which is then passed unaltered into the next generation with mutant offspring derived from this parent making up the remainder of the population. Alternatively, the user can evolve the population by crossover. Here, the user selects two or more parents and the next generation is entirely filled by their offspring (as described above). When more than two parents are selected for crossover, each offspring is created by crossing two of those parents (chosen randomly). Finally, the user can choose mutation and crossover in which each offspring is either created by mutation or by crossover, the probability of which is determined by the lowest slider in Figure 6-1. At any point, the user may select a tile for editing in which they can freeze the shape, size and color of any (or all) of the tile's components (see Figure 6-4).

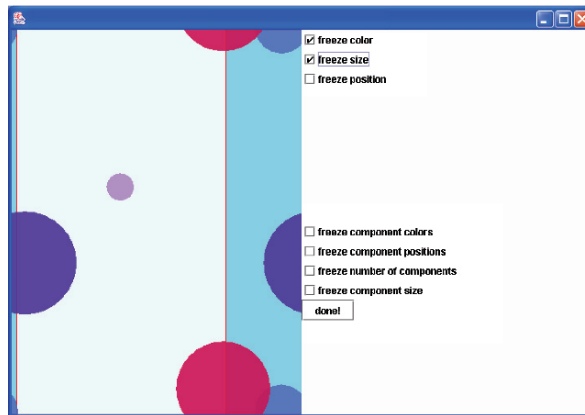


Figure 6-4. The bottom right tile in figure 1 is edited: here the vertical stripe is selected and its color and size frozen.

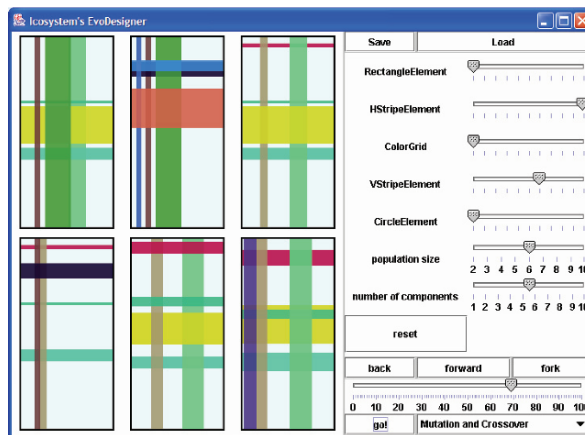


Figure 6-5. Example plaid tiles with a relatively high proportion of vertical and horizontal stripes.

3.1.1 Initialization. Initialization occurs by either loading a population of tiles from a file, or by randomizing the population. A slider determines the number of graphical components within a tile and other sliders determine the relative proportion of the different graphic component types. Thus, one can easily increase the absolute number and relative proportion of vertical and horizontal stripes to explore plaid patterns (see Figures 6-5 and 6-7) or circles for more retro designs (see Figures 6-6 and 6-7).

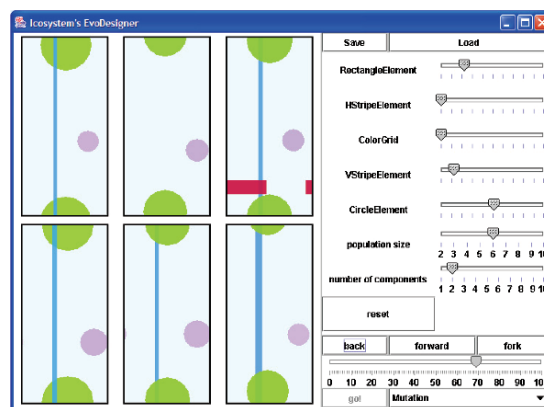


Figure 6-6. A retro pattern with a low number of graphical components but high proportion of circles.

3.1.2 Exploring Design Space. In interactive evolution the user usually does not know in advance quite what they are after; it's often a case of "I'll know it when I see it." However, as in any exploratory activity, one can be led down a blind, unfruitful alley. For this reason, the *evoDesign* tool includes a back button that allows one to revisit previous generations. Thus, one can explore one avenue, decide that this is not a desirable portion of the design space and return to an ancestor population of designs to explore other avenues using the mutation and crossover operators as before. Additionally, the save/load feature

can be used with a similar effect and there is also a fork button that allows one to spawn off a new instance of the tool with the same current design population. This ability to revisit any point in the current run’s “design tree” is particularly useful and important; without it, it can be surprisingly frustrating when a design is not evolving satisfactorily. Despite the seemingly limited number of graphical components, one can evolve a surprising variety of pleasing design (something one *would* want to wear to work or tile a bathroom with), and ones that represent or evoke different styles and decades. A few “samples” are shown in Figure 6-7.

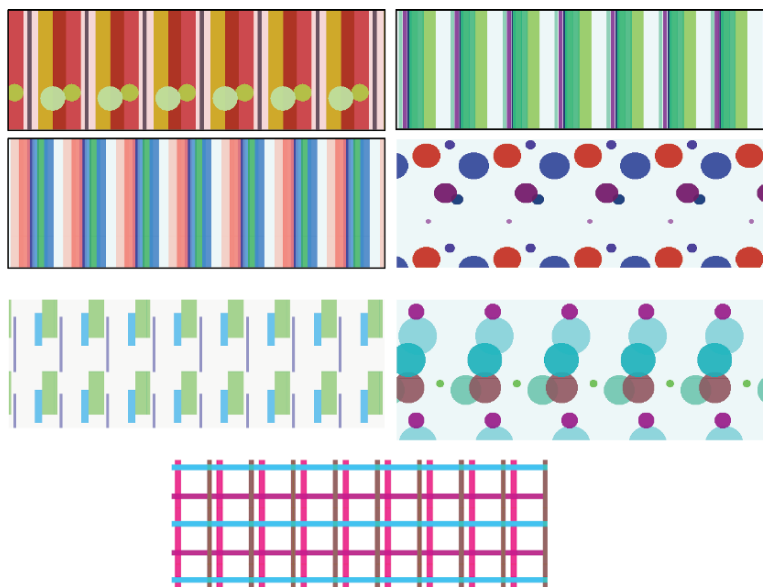


Figure 6-7. Some evolved wallpaper or fabric “samples”.

O’Reilly and Testa (unpublished), in the context of interactive evolution, suggest that “architects are interested in families of designs rather than finding one absolute best.” Similarly, other designers are interested in designing a set of pieces which together form a coherent collection. Each new piece has to conform to that collection’s or designer’s signature style but it is hard to define precisely what that is. *evoDesign* can be used to such effect. Figure 6-8 shows

a target design, Paul Smith stripes, and a population from evoDesign that has a similar Paul smith look and feel.

4. DISCUSSIONS

Interactive evolution (using GAs) has been used in a number of recent 2-D graphic-related applications, including evolving color schemes (Kelly 1999), photofit composite images (Frowd et al. 2004), anthropomorphic symbol designs such as a warning sign (Dorris et al. 2003), and product branding and marketing (see affinova.com). However, the full potential of interactive evolutionary creative design has yet to be realized.

What we set out to do was to create a tool that was intuitive, useful and practical. The result happened to be a system that is significantly simpler to understand and easier to use. The target users of our case study were design experts because they would not only use such a tool more frequently than the general populace but because they also have the most to give: the system gets the user's expert esthetic and design expertise to judge "fitness" while the user gets the ability to explore around a design theme, within constraints (see Figure 6-8).

This simplicity, however, has profound implications not just for selling the idea to the experts—here, non-technical design experts—but also for ultimately making interactive evolution a commonly used technology in the general populace. There is almost no better way to promote a technology than to simplify it for the masses and give it away for (almost) free. Thus, Apple Computer's bundling of GarageBand in their iLife package has essentially commoditized music editing and recording. This has yet to be achieved with interactive creative design because previous systems are too application specific (and it is difficult to demonstrate the broader implications of the technology) and/or too complex for the majority of general users (non-computer scientists) to understand. While the general public doesn't need to understand all of the gears "under the hood" to be able to use the tool, it is helpful to provide them with the general information of the implications of their actions within the tool. In the case of evoDesign tool, the key is its ability to do genotype-phenotype mapping. In genetic art, most design tools are based on genetic programming, where the genotype of

compact, abstract expression bears no resemblance to the final picture—the phenotype. In a simple genetic algorithm, such as that used in *evoDesign*, the mapping is much closer: what is being evolved and how those changes in the genotype affect the phenotype is much more closely related and easier to explain and to understand. Moreover, in order to be able to edit components of the design directly in the GUI and have those changes reflected in the genotype, it is crucial that the genotype mapping is 1:1 and completely reversible (O’Reilly and Testa, unpublished). The simpler the genotype, the more likely this is to happen.



Figure 6-8. A target pattern: Paul Smith stripes and a population from the *evoDesign* tool.

4.1 An Ideal Interactive Evolution Tool

evoDesign is by no means perfect, but we believe is headed in the right direction, at least according to guidelines set out by Bentley & O’Reilly (2001) in their “ten steps to make a perfect creative evolutionary design system.”

We have a specific domain ‘in which it makes sense to use a computer for “creativity enhancement” ’ (their rule 1). We have a “good reason for using a creative system at all” (rule 2). As discussed, a GA design tool can serve several sensible purposes: to generate new ideas or to explore the design space around a seed idea, and to do so in a manner that essentially mirrors the way that a designer might modify a design.

We believe that we have “appropriately balanced control of the design process between the tool’s user and the tool itself” (rule 3)—the ability to freeze elements, to choose particular parents for crossover, to set the mutation rates and other parameters are all under the control of the users. However, a user-friendly editor in which a designer could create a specific initial seed tile, to interact with and modify *all* parameters of elements, and even to drag and drop elements, would be even better. The input and output format for the saved designs can be very easily tailored to systems that professional graphic designers use (rule 4). We hope we have demonstrated that evoDesign is “generative and creative” (rule 5) and “understandable” (rule 6). Because it is based on esthetic choice, and because of the ability to revisit designs in the previous generations, there is certainly “an easy and effective way of evaluating the quality of solutions and guiding the path of evolution” (rule 7). We are currently working on rule 8: “find people who are actually prepared to use the system”. For completeness, we list the rest of their rules here: rule 9 is “get lots of money to pay R&D costs” and rule 10 is “start a company and make a billion.”

4.2 Constrained Designs

In evoDesign, we included some simple design constraints such as the minimum and maximum width of stripes, the maximum radius of circles and so on. However, many designers are faced with more complex and subtle constraints; for instance, a client may have a particular defining color palette (say the red/white/blue/black of *Tommy Hilfiger*) or “feel and style”—e.g., *Versace* has a very definite style but is hard to define; again, this is a case of “I know it when I see it”—and the designer may only explore color-“equivalent” designs (*sensu* Feldman (2005)). Or, the design may need to evoke a particular era, say the bright colors of the 1960’s or earthy tones of the 1970’s. Such constraints can be incorporated into the GA. Kelly (2004) describes a GA-based applica-

tion that he used to evolve color schemes. He explored several different types of constraints (Eckert et al. 1999), one of which is to average the color combinations in the scheme such that the colors in the design may evolve but under the constraint of the overall constraint of “greenishness”.

Another important area of constraint, one that is not necessarily obvious but can be very important, is a design’s manufacturability. Not all designs are alike: some designs are far easier and cheaper to produce than others, either because of the specific material required for a particular color whose color mixing gives the rate at which a product can be manufactured by a given machine, or there is a limit on the number of colors available for a machine such as a loom. However, some of these features can be subtle and not necessarily obvious in the final design—one may be able to make the design easier and cheaper to produce but have very little effect on the overall appearance and esthetic appeal, at least to the casual observer. Thus, it is possible to associate each design with a number of metrics, displayed in the user interface (Bandte and Malinchik 2004), and the user can then evolve designs that are both esthetically pleasing, satisfying esthetic design constraints (such as greenishness), and are also easy to manufacture.

4.3 Wallpapers

Our tiling feature (Figure 6-2) is a regular, rectangular lattice tiling. However, there are other ways in which one can tile a wall: just sixteen other ways in fact. Considering all the combinations of translation, reflection and rotation of a tile, mathematically speaking, there are only 17 types of wallpaper (“wallpaper groups”). For instance, one can place the tiles on a rectangular or hexagonal lattice, keeping all tiles in the same rotation (as in Figure 6-2) or rotate alternate tiles by 180° to form a checkerboard arrangement where white and black would represent tiles with different orientations and so on. What would these other wallpaper types mean for a tool such as *evoDesign*?

First, one would have to remove the toroidal nature of the tiles. That is, we wrapped the design around each tile so that they would be guaranteed to tile. However, while this works for this most basic wallpaper type, it does not hold

true for all other wallpaper types. Let's consider the checkerboard arrangement. A tile that contains just one circle that overlaps the upper left margin of the tile would be positioned on the *lower right* margin of the adjacent tile; in short, they would not match up and there would no complete circles anywhere in the design. Thus, as a general rule across all wallpaper types, wrapping must be removed.

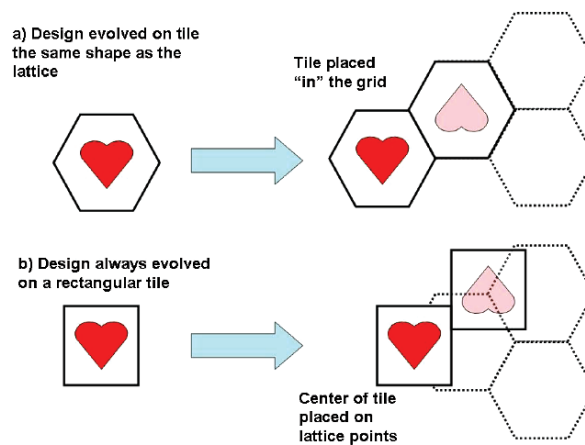


Figure 6-9. Two approaches to evolving tiles for the 17 different wallpaper types. a) evolving a tile of the same shape as the underlying lattice. b) evolving a rectangular tile and placing it on the lattice points.

Second, the tile shape could be altered. The underlying lattice in these 17 plane symmetry groups are parallelogramic and rhombic (the latter being a special case of the former thus they are essentially the same), rectangular and square (*ibid.*), and hexagonal. One approach would be to evolve designs on tiles of these different shapes; e.g. the panel would be a hexagram (see Figure 6-9a). Another approach would be to evolve designs on an unwrapped rectangle and then the center of that design would act as the pivot point of the tile and the placement point on the lattice (see Figure 6-9b). That is, given a lattice, one places the tiles in the grid (first approach), or place the tiles on the lattice points of that grid (see Figure 6-9). Mathematically, they are the same (if the lattice is regular). However, from a practical perspective they have different pros and

cons. In the first approach, the interface would change as the wallpaper type changes but the same tile cannot be used in different wallpaper types, something that one might want to evolve in the tool. In the second approach, the interface and tile shape is always the same but it is harder to judge and control the overlap between tiles which can radically affect the design. When reflection and rotation are involved, it is hard to determine how the final wallpaper will look from a given tile.

However, there are existing tools (such as Artlandia's SymmetryWorks® (artlandia.com)) in which one is able to draw a shape on screen and immediately see that shape tiled into a wallpaper. Then one can move and otherwise modify any of the shapes in the wallpaper and those changes will immediately be reflected in the whole wallpaper. In the case of fabrics and wallpapers—importantly, where the end goal is a *tessellation* rather than a tile—a very powerful tool could be created by marrying such a wallpaper design tool with an evolutionary tool such as evoDesign whereby one would evolve whole fabric or wallpaper design at once rather than individual tiles.

References

- Bandte, O., and Malinchik, S. A (2004) broad and narrow approach to interactive evolutionary design—an aircraft design example. In GECCO 2004 (K. Deb et al., eds), LNCS 3103, pp. 883–895. Springer Verlag, Berlin.
- Bentley, P. J., and O'Reilly, U.M. (2001) Ten steps to make a perfect creative evolutionary design system. In the GECCO 2001 Workshop on Non-Routine Design with Evolutionary Systems.
- Bentley, P. J. Aspects of Evolutionary Design by Computers. In *Advances in Soft Computing - Engineering Design and Manufacturing*, Springer-Verlag, London, 1999, pp. 99–118.
- Dorris, N., Carnahan, B., Orsini, L., and Kuntz, L.-A. (2000) Interactive evolutionary design of anthropomorphic symbols. In *Proceedings of Congress on Evolutionary Computing (CEC'04)* (Portland, OR, June 19–23, 2003) IEEE, 433–440.
- Eckert, C., Kelly, I., and Stacey, M (1999) Interactive generative systems for conceptual design: an empirical perspective. In *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 13, 303–320.
- Feldman, U. (2005) Quantifying the dimensions of color experience. Unpublished Ph.D. thesis. Massachusetts Institute of Technology.
- Frowd, C.D., Hancock, P.J.B., and Carson, D. (2004) EvoFIT: A Holistic, Evolutionary Facial Imaging Technique for Creating Composites. *ACM TAP*, Vol. 1 (1), pp. 1–21.
- Kelly, I. (1999) An evolutionary interaction approach to computer aided colour design. Ph.D. Thesis, The Open University, Milton Keynes, UK.

- McCormack, J. (1993) Interactive evolution of L-systems grammars for computer graphics modeling. In *Complex Systems: from Biology to Computation* (D. Green, T. Bossomaier, eds) ISO Press, Amsterdam.
- Rooke, S. (2005) The evolutionary art of Steven Rooke. <http://www.dakotacom.net/~srooke/index.html> (Jan 7, 2005).
- Sims, K. (1991) Artificial evolution for computer graphics. *ACM Trans. Comput. Graphics* 25, 319–328.
- Takagi, H. (2000) Active user intervention in an EC search. In *5th Joint conference on Information Sciences (JCIS2000)*, Atlantic City, NJ.
- Takagi, H. (2001) Interactive evolutionary computation. In *Proceedings of the IEEE* 89(9), 1275–1296.
- Takagi, H. (1998) Interactive evolutionary computation—cooperation of computational intelligence and KANSEI. In *Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA '98)*, World Scientific, 41.
- Todd, S., and Latham, W. (1992) *Evolutionary Art and Computers*. Academic Press.
- Whitelaw, R. (1999) Breeding aesthetic objects: art and artificial evolution. In *Creative Evolutionary Systems* (edited by Bentley, P.J. and Corne, D.W). Morgan Kaufman. 129–145.
- Whitelaw, R. (2004) *Metacreation: Art and Artificial Life*. Cambridge, Mass.: MIT Press.