Evolutionary reproduction of Dutch masters: the Mondriaan and Escher evolvers

A.E. Eiben[1]

Free University Amsterdam `gusz@cs.vu.nl`

1 Abstract

This chapter discusses two applications aiming at evolving images in the styles of two well-known Dutch painters: Mondriaan and Escher. For both cases we have an evaluation criterion based on "style-fidelity". This makes the (subjective) selection less free than in applications solely aiming at nice images. Technically, the Mondriaan evolver is less difficult, given that his style "simply" uses rectangles filled with a single color. The Escher evolver project is more challenging. First, because Eschers style is less simple to capture. Second, the system is to be implemented *in vivo*, in a real museum, posing requirements on the interface. We describe how to meet the style challenge based on the mathematical system behind Eschers tiling. Designing a suitable representation and the corresponding variation operators based on this system specifies an appropriate search space guaranteeing the Escher style to some extent and leaving enough freedom for the selection. As for the second objective, we describe two versions, that differ in the way the images are presented to, respectively evaluated by the visitors. The experiences gained during a six month exhibition period in the City Museum in The Hague, Netherlands, are discussed from the visitors perspective as well as from the algorithmic point of view and are illustrated with some "evolved Eschers".

2 Introduction

A fundamental question raised by evolutionary art projects is: Who is creative here? Is it the user who executes (subjective) selection, or is it the evolution that produces the items to be selected by means of reproduction? In general, for any evolutionary process one could say that reproduction is the power pushing novelty –by creating new stuff– and selection is the force behind quality –by propagating good stuff [4]. In these terms the question is whether selection or reproduction is the creative force, or, whether novelty or quality

accounts (more) for the creative experience of the users. The work described here does not give an answer to this, but rather, it indicates another likely source of creativity: the applied representation.

3 The Mondriaan Evolver

P.C. Mondriaan is considered one of the most prominent 20th century geometric painters. In 1917 he and three others founded the journal "De Stijl" wherein Mondriaan published twelve chapters on his vision on new art. Around that time he started painting only in abstract and some years later he took this style one step further by using only primary colors and straight black lines that intersect at right angles.

The Mondriaan project is rooted in the authors university course on evolutionary computation including a programming assignment: implementation of an evolutionary system that creates images in the style of Mondriaan. The first version of such a Mondriaan Evolver has been implemented together with J. van Hemert [8], later adjusted by B. Craenen[1]. As for representing Mondriaan-like images (the phenotypes) by simple code (the genotypes), there are numerous simple options. In the first version of our system chromosomes consist of two parts, one standing for the color and one for the composition. The decoding (genotype - phenotype mapping) is done by a recursive function. This function takes the canvas and starts dividing it up into smaller planes using the data in the chromosomes to decide where to split and what colors to use for the resulting planes. Such a straightforward linear chromosome structure allows using simple genetic operators: single point crossover and n-point mutation. Alternatively, a tree-representation can be used as well. Within such a tree an internal node specifies a split of the rectangle by the nodes label (horizontal/vertical) and its three child nodes that determine the position of the line to split and the colors of the resulting two subplanes. Instead of a color, a child can be another internal node, leading to a simple recursive system. Figure 1 illustrates this system.

As for selection, it takes place through a GUI presenting 9 images to the user who can grade them on a given scale. This can be 1 to 10, or in later versions 1 to 3: good, neutral, bad. This GUI is illustrated in Figure 2, showing a screenshot from the on-line Mondriaan Evolver.[2]

4 The Escher Evolver

The project described here emerged from a cooperation with the City Museum in The Hague, The Netherlands, that intended to organize a large exhibition

---

[1]http://www.xs4all.nl/ bcraenen/EArt.html
[2]http://www.cs.vu.nl/ci/Mondriaan/Mondriaan.html

Fig. 1. Tree-based representation of Mondriaan style images

Fig. 2. Graphical user interface of the Mondriaan evolver

devoted to the works of M.C. Escher [3]. Although Escher had no background in mathematics, his insights and self-study led him to an art form that is based on the geometry of two and three dimensional spaces. Very popular are his pictures based on tiling of the two dimensional plane, [5, 7]. These images feature animal like figures that complement each other thus forming a complete coverage of the plane. Through informal contacts between the university and the museum the idea emerged that the traditional Escher exhibition should be extended by a virtual part: computer generated images shown to visitors on LCD screens hanging on the walls among the original works of the artist. The essential aspects of the idea have been the following.

1. The computer generated images should be in the style of Eschers tilings.
2. Images should be generated by an evolutionary process, that is, random variation and fitness-based selection on a population of images.

3. The visitors should be able to control the evolution by subjective selection.
   That is, looking at the pictures they could vote on the images and thereby
   set the fitness values, defined as the balance between votes in favour of
   and against of a given picture.

The key to meet the first requirement was the mathematical system behind
Eschers tilings.

5 Eschers tilings: the mathematical system

A nice overview of the mathematical system behind Eschers tiling is given in
[7], pg. 31-69. Here we briefly recap the most important aspects of it. Escher
used two different ground shapes when creating his tessellations: triangles or
parallelograms. Our evolutionary program uses only parallelograms. Includ-
ing special cases of the parallelogram we have four different ground shapes:
the parallelogram, the rhombus, the rectangle and the square. To fill a plane
with one of these ground shapes one can use three different transformations:
translation, rotation and glide reflection (a combination of a reflection and a
translation). Using these transformations one can create 10 different transfor-
mation systems for filling the plane. These are numbered traditionally with
Roman numbers as I, II, ..., X. During this project we did not consider trans-
formation system X since it operates on triangles. A few translation systems
work on all ground shapes, the others assume equal side length (rhombus or
square) or square corners (rectangle or square). Table 1 shows all possible
combinations of ground shapes and systems after [7], pg. 58. All of these sys-
tems are repeating, that is, we can always create a block of these shapes that
can fill the plane using only translation.

Table 1. Possible combinations of shapes and transformations

	I	II	III	IV	V	VI	VII	VIII	IX
A. Parallelogram									
B. Rhombus									
C. Rectangle									
D. Square									

The left hand side of Figure 3 shows the details for system II using a simple
rotation. The right hand side illustrates system IV that uses 2 glide-reflections.
System VI (not illustrated here) is the most complicated, 2 rotations and 2
glide-reflections are used in such a way that only a block of 16 shapes is large
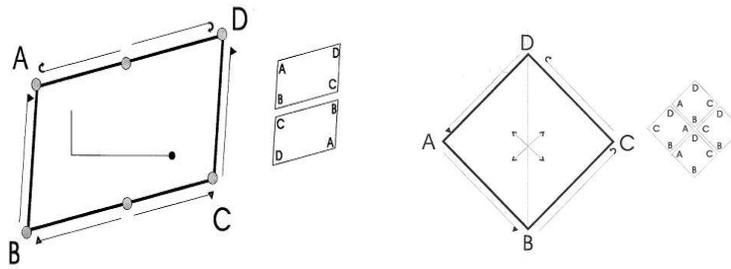enough to fill the plane by translation only.

Fig. 3. Transformation systems II (left) and IV (right).

6 Evolvable representation of Eschers tilings

The computational engine behind the Escher Evolver is an evolutionary algorithm. In the whole algorithm design it is the representation that forms the greatest challenge, because it determines the syntax (set of genotypes) and its corresponding semantics (set of images encoded by these genotypes). Thus ultimately it determines what kind of pictures can evolve at all. The other components of the evolutionary system, genetic operators, parent selection, and population update rule (survivor selection), are more straightforward.

The algorithm operates on a genotype consisting of an array of integers. Several steps are needed to transform these into an Escher-like image. The first step in generating an image is calculating image properties such as size, shape and colours using the genetic information. These are used to create several affine transformations used to transform and paint the building blocks, i.e., the tiles forming the basic units of the final image, onto our drawing canvas. Second, the curves forming the contour of a single building block are calculated, then the filling of the block is done. We then have a vector image of a single building block. Using the affine transformations this is extended to a tiling and transformed into a bitmap before being drawn onto the canvas. Next we take a detailed look at these steps beginning with transforming the genotype into image properties. Table 2 shows the genes we use and the properties they determine.

6.1 Ground Shape and Transformation System

In our genotype we store the ground shape and translation system. Both can be changed independently during crossover and mutation. Therefore we apply a repair algorithm after the reproduction phase. This algorithm reverts a invalid combination to a neighbouring valid combination. The vertex displacement genes were to be used for distorting the image by moving the cornerpoints of the shapes, but weve never implemented these displacements. Escher has used them in numerous images though.

Table 2. Escher representation. For more details, see the corresponding subsections.

Gene no.	Trait	Converted into range	Note
GROUND SHAPE			
0	Ground Shape	A,B,C,D	
1	Ground Shape angle	amin . . . 180-amin	A and B only
2	Edge-Ratio	emin . . . emax	A and C only
3	Transformation System	1 . . . 10	
4. . .5	Vertex Displacement		Unused
PLANE TRANSFORMATION			
6	Plane Scaling factor	smin smax	
7	Plane Rotation Angle	0 . . . 359	
8. . .9	Old color		Unused
FILLING OF THE SHAPE			
10	Type of view	Top / side view	
11	Head Corner	1 . . . 4	
12	Eye distance	0.1 . . . 0.3	
13	Mouth type	none, loose, beak	Side view only
14	Wing type	none or one of 4 types	
15	Number of wing lines	2. . . 5	
16	Wing width	0.2. . . 0.5	
17	Wing length	0.3. . .0.8	
18	Tail type	none, full	
19	Number of tail lines	3 . . . 6	
20	Tail edge length	0.2 . . . 0.5	
21	Tail center length	0.2 . . . 0.5	
22	Spine	true, false	Top view only
COLORS			
23. . .25	Foreground color	HSB-color	
26. . .28	Background color	HSB-color	
CURVES			
29. . .38	curve 1	3rd-order Bezier curve	
39. . .48	curve 2	3rd-order Bezier curve	
49. . .58	curve 3	3rd-order Bezier curve	
59. . .68	curve 4	3rd-order Bezier curve	

6.2 Curves

The contour of a shape is build up out of 2, 3, or 4 different curves. How many curves are needed, where each curve is placed and which direction they are facing is determined by the transformation system in use. For each of the nine systems we have separately encoded this into the drawing algorithm. To represent the curves we use high-order Bezier curves (see Figure 4) because we can use as many control points a necessary and Bezier curves have a convex nature, which assures our curves do not cross each other. In the final implementation 5 control points are used.
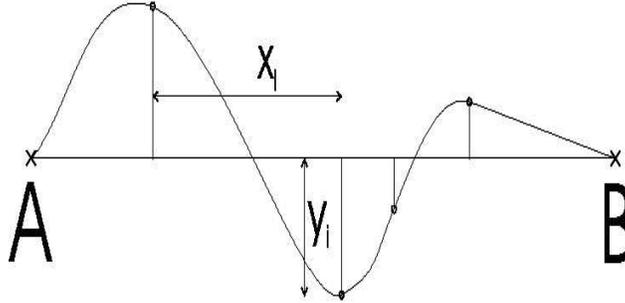
Fig. 4. Representation of a curve

6.3 Filling the shape

Use only a contour, the drawing will never look like an animal such as a bird, fish or reptile. We have to use eyes, wings and tails for this. First we tried to use curves for the fillings to, assuming the evolutionary algorithm would choose the curve that make the figure look alive. This didnt work very well. So we chose a set of predefined curves to fill the shapes. First we have a bit that decides whether the picture is a side-view (birds, fish) or top-view (fish, reptiles). The top-view has 2 eyes and possibly a spine. The side view has one eye only and can have a mouth. Both types can have several types of wings. The left-hand side of Figure 5 shows a diagonal wing. The direction, number and length of winglines are genetically determined. The same applies for the tail section. Here the curve closing the tail, and the number of tail lines are drawn using several genes. The right-hand side of Figure 5 displays a typical tail configuration. Before drawing, the entire filling is clipped to the shape area, so the wings and tail lines always end at the contour.

6.4 Colours

The image has 2 colours, a foreground and a background colour. Both are represented in a HSB (Hue-Saturation-Brightness) colour model. Not all values
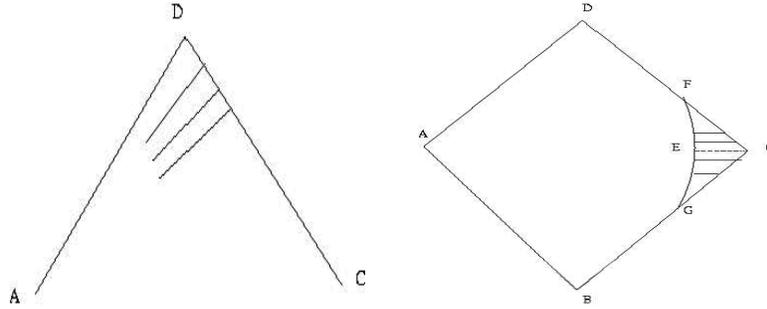
Fig. 5. Illustrations of filling a shape: diagonal wing (left) and tail (right).

for H, S and B are possible, for example the foreground colour is a little bit brighter than the background colour. The old colour genes (one integer per colour) are not used anymore. They produced not enough different colours.

6.5 Plane Transformation

After the entire image has been made in vector format it needs to be transformed into a bitmap. The plane scaling factor multiplied by a target dependent factor form the size (in pixels) of one shape. Figure 6 shows an image obtained through this representation. The one showed here was named "The flatfish".
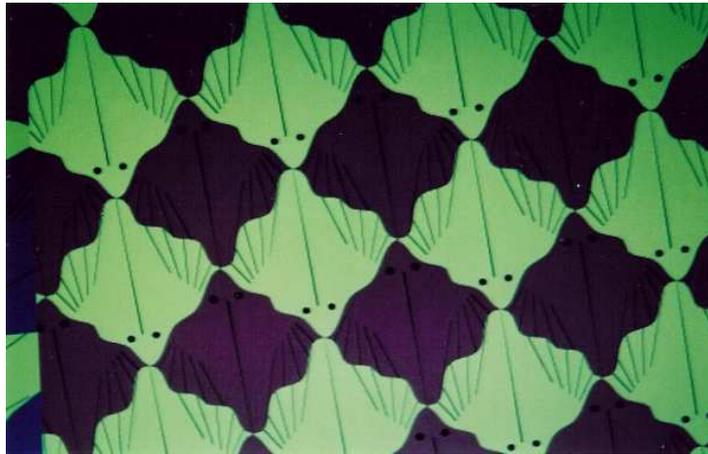
Fig. 6. Illustration of a possible image phenotype: The flatfish

7 Reproduction and selection operators

The initial evolutionary algorithm was based on simple operators: one-point crossover, random change mutation, tournament selection, and generational replacement. After having gained experience with the resulting system we had to conclude none of these choices was the right one, and we chose alternatives for all of them. We briefly discuss the four main operators below.

N-point crossover One-point crossover did not satisfy our needs; for instance the two old colour genes were located next to each other. Because of this the offspring often had both colours of one single parent. Using uniform crossover, the curves did not cross very well. After a number of comparative experiments we found that N-point crossover (with a random number of crossover points) provides a better mixture of genes, keeping parts of the curve structure intact.

Gaussian mutation When we used replacement of genes with random values, strange mutations occurred. For example, curves became totally different, or radical colour shifts and huge size variations occurred. For users (experimenters) such changes were too big, the process looked rather a random walk than gradual evolution. A more subtle approach was required, where a gene was not replaced by a random value, but was mutated by an increment. The value for this increment was chosen from a Gaussian distribution with mean $= 0$ and $0.2 \cdot gene_{max}$ as standard deviation. This value (positive or negative) is then added to the existing gene value.

Roulette wheel selection To perform parent selection we first applied tournament selection as used in the Mondriaan Evolver. However, experiments during development indicated that it did not work as expected. Simple roulette wheel selection worked well, and was therefore chosen for the final implementation.

Steady-state replacement As for the survivor selection mechanism, we started the exhibition period with using a generational model, where all present individuals die and are replaced by the offspring. Later on we changed to a steady-state model (only replacing part of the actual population), see next section.

8 Working of the system

Two implementations of the above algorithm have been at exhibition in the City Museum The Hague. Both are networked systems that use flat LCD screens to gain votes, based on the same stand-alone version. The main differences between the first and second version are the evolutionary model used (generational vs. steady state) and the way the visitors vote (either for/against a single picture or a choice between two given pictures).

In the stand-alone version the whole population of six images is shown to one user in one screen. The evolutionary mechanism is completely hidden, (that is, the user cannot experiment with different mutation rates, or various crossover types) all the user needs to do is to evaluate the images and press the button for the next generation. This way, the user can quickly see how to direct the evolutionary process to a certain direction. We used this version to test the genetic decoder and to compare different algorithmic setups.

8.1 First networked version

The first networked version was actually a copy of the stand-alone version. Here, six flat screens have been hanged on the walls of the museum, among the regular works and one image was shown on each screen. Note that hereby, the population size was limited to the number of screens (six), which resulted in a rather small population. Based on this setup a collective of visitors, rather than one user, were evaluating the images, thereby delivering the necessary votes to compute the fitness values of the pictures. Physically, each screen was connected to a client PC, and these clients were all connected to a server.

The work was divided between the server and the clients. The server actually runs the genetic algorithm and it sends the genetic information to the clients, one individual to each client. The client decodes the genotype it receives and shows it on the screen. The visitors can use 2 buttons under the screen to vote "yes" or "no" (like this picture or do not like this picture). At a regular interval each client sends its vote count to the server. After an interval between 5 and 30 minutes (depending on the number of votes received) the server counts all the votes and calculates the fitness of each individual. This is simply the percentage of "yes" votes using some bias to handle the situation of no votes at all. Finally the server performs an evolutionary cycle (parent selection, reproduction, survivor selection) to calculate the next generation, which replaces the current one and is sent to the clients again.

Although the software worked fine, the evolution did not work well. There were not enough votes and there was much convergence, i.e. very similar pictures in the population. The causes were found in the user interface and algorithm setup:

1. The population size of 6 was to small to hold enough diversity.
2. Sometimes –even when running on 30 minute intervals– the total number of votes was only 20 per interval. This resulted in individuals that became very dominant in one or two generations.
3. Visitors thought the voting instructions were unclear. Some interpreted them as "Do you like the picture?", others as "Is the picture Escher-like?". Furthermore, it turned out that people would prefer voting in the context of other pictures, using picture(s) A (B,C, ..) as a reference to judge picture X. This was impossible because of the distribution of the screens in different rooms in the museum.

4. When a visitor voted, he or she only got a "thanks for your vote" message as feedback on the screen. This was neither amusing nor interesting enough to motivate them to vote at other screens too. Apparently, people expect an immediate effect after pushing a button.

5. The variance in colour was not sufficient, we had only light foreground and dark background colours.

All these factors together made the screens and the whole virtual exhibition not attractive enough. Therefore, a second version was implemented to overcome these problems, while still using the same hardware and genetic representation.

8.2 Second networked version

Apart from introducing the new colour system, which uses 3 genes per colour instead of one, the genetic structure was unchanged. The main novelties were in the reproduction mechanism and the user interface.

We had to increase both the population size and the number of votes received, but we could only use six flatscreens. To accomplish the larger population we allocated 5 individuals to each flatscreen, which made a total of 30 individuals in the population. We could not generate 30 new individuals each generation, we had not enough votes. A steady state algorithm –where not the whole population is replaced in one cycle– could solve the problem, because an individual that stays in the population for many (approx. 5-10) cycles can get much more votes. In this model the raw fitness of an individual is calculated just as in the old model. The fitness values used for reproduction and survival are calculated using age-dependent transformations of the raw fitness values. This protects young individuals from dying quickly and make old ones reproduce less and die faster. The age correction for reproduction is as follows. The fitness of an individual that has been around in the population is transformed to lie between zero and L_r, where L_r is an age-dependent limit. L_r is 100 for individuals two generations old, and reduces by 10 for each age up to age seven and by a decrement of 20 afterwards. After age nine $L_r = 0$. Using roulette wheel selection based on this age-corrected fitness values means that only individuals between two and nine generations old will ever be selected as parents for reproduction. The correspondence between fitness and survival in the population is treated differently, as follows. Individuals of zero (newborn) or one generation old are given maximum fitness $L_s = 100$. The fitness of an individual two generations old is scaled between $L_s = 60$ and $L_s = 100$, that is, ensuring a survival fitness of at least 60. Individuals from five to nine generations get constrained by a maximum of $L_s = 90, 80, 70, 50$, and 30, respectively, and individuals after nine generations have $L_s = 0$.

The user interface now has the task of accumulating votes for different individuals. It manages this task by showing two pictures at the same time on a split screen, see Figure 7. The user can vote in favour of one of them. Note

that the meaning of the two buttons under the screen changed from yes/no for one picture to yes for left/right picture. Each time a vote has been made, the client thanks the visitor by replacing the looser with an instruction text. A few seconds later two new pictures, drawn randomly from the pool of 5 individuals available to the client, are shown. The user may then vote again

Fig. 7. Split screen in the second version of the system in the City Museum, The Hague. Viewers press the buttons below the display to indicate their preferred image.

The pictures that have emerged in the museum during the approximately thousand generations do show resemblance with Eschers tilings, but they could never be mistaken for a real Escher. We conjecture that there were two factors that prevented the evolution of convincing Escher-like images. One lies in the subjective selection of the ever changing set of visitors. The collective intelligence, or rather, the collective taste, was arguably not consequent enough to direct the process towards a specific type of image. This effect would obviously not occur in a stand-alone system controlled by a single user. The other one is rooted in the limitations of the representation. This could only be helped by designing more complex genotypes and corresponding decoders.

The exhibition has been visited by thousands of people, who did actively participate in an evolutionary process of breeding art. Even though the "style-fidelity" of the images was limited the visitors reactions were mainly positive. An interesting aspect was the cognitive experience of the visitors. As discussed above we noticed a preference for comparing images, as opposed to simply voting for/against one picture. Nevertheless, the overall evaluation of the museum management was very positive; the combination of traditional art and new media, together with the active involvement of the visitors in creating the art shown, have been greatly appreciated.

9 Concluding remarks

Systems using interactive evolution, i.e., evolutionary algorithms with subjective user selection, have proven capable of achieving creativity, cf. [1, 2]. It can be argued that in such systems the source of creativity is the user, respectively, the evolutionary reproduction mechanism. The experiences with mimicking existing artists shed new light on this question. Paraphrasing Koza *et al.* [6] one could say that the ultimate challenge to a creative evolutionary system is to be human-competitive by producing the same quality. Then, mimicking existing human artists forms a canonical challenge, not unlike the Turing test for intelligence.

In this chapter we described two projects aiming at evolutionary reproduction of images of existing artists, Mondriaan and Escher. Of these two, the Mondriaan showcase was easier. The reason is that the underlying structure (straight lines and primary colours) is easier to represent in a computer. The Escher Evolver –although it can create impressive images– needs more sophisticated representations to be really convincing. These experiences suggest that the representation is one of the most essential sources of creativity. For the time being, imitating existing art by evolutionary processes remains a great challenge.

Acknowledgements

References

1. P.J. Bentley, editor. *Evolutionary Design by Computers.* Morgan Kaufmann, 1999.
2. P.J. Bentley and D.W. Corne, editors. *Creative Evolutionary Systems.* Academic Press, 2002.
3. A.E. Eiben, R. Nabuurs, and I. Booi. The Escher evolver: Evolution to the people. In P.J. Bentley and D.W. Corne, editors, *Creative Evolutionary Systems*, pages 425–439. Academic Press, 2002.
4. A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003.
5. M.C. Escher and F Bool. *Regular divisions of the plane at the Haags Gemeentemuseum.* Haags Gemeentemuseum, 1986.
6. J.R. Koza, M.A. Keane, J. Yu, F.H. Bennett, and W. Mydlowec. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1(1):121–164, 2000.
7. D. Schattschneider. *Visions of Symmetry: Notebooks, Periodic Drawings, and Related Work of M.C. Escher.* W.H. Freeman and Company, 1990.

8. J.I. van Hemert and A.E. Eiben. Mondriaan art by evolution. In E. Postma and M. Gyssens, editors, *Proceedings on the Eleventh Belgium-Nederlands Confererence on Artificial Intelligence (BNAIC'99)*, pages 291–293. AI, 1999.