An artist’s experience in using an evolutionary algorithm to produce an animated artwork

Karen Trist*, Vic Ciesielski and Perry Barile

RMIT University, G.P.O. Box 2476V, Melbourne, Victoria 3001, Australia
E-mail: karen.trist@rmit.edu.au
E-mail: vic.ciesielski@rmit.edu.au
E-mail: perry.barile@rmit.edu.au
*Corresponding author

Abstract: We describe an artist’s journey of working with an evolutionary algorithm to create an artwork suitable for exhibition in a gallery. Software based on the evolutionary algorithm produces animations which engage the viewer with a target image slowly emerging from a random collection of greyscale lines. The artwork consists of a grid of movies of eucalyptus tree targets. Each movie resolves with different aesthetic qualities, tempo and energy. The artist exercises creative control by choice of target and values for evolutionary and drawing parameters.

Keywords: evolutionary programming; evolved art; new media art; software art; animation; algorithmic art; AI; genetic programming; genetic art; NPR; non-photorealistic rendering.

Reference to this paper should be made as follows: Trist, K., Ciesielski, V. and Barile, P. (2011) ‘An artist’s experience in using an evolutionary algorithm to produce an animated artwork’, Int. J. Arts and Technology, Vol. 4, No. 2, pp.155–167.

Biographical notes: Karen Trist is a Lecturer in Photography and Postgraduate Supervisor in the School of Media and Communication at RMIT University. She is also the Program Director for the BA Creative Media Honours. She gained her MA in Photography from RMIT in 1999. Her research interests in the field of photography encompass issues pertinent to globalism and sustainable cities, cross-cultural and gender studies. She has been part of the evolutionary art group at RMIT for four years. Her photographic work has been published extensively worldwide and she has exhibited her work in galleries, both in Australia and internationally.

Vic Ciesielski is an Associate Professor of Artificial Intelligence in the School of Computer Science and Information Technology at RMIT University. He received his BSc from the University of Melbourne and his PhD from Rutgers University. He is the Leader of the RMIT Evolved Art Group. In addition to evolved art, his research interests include evolutionary computing, genetic programming and evolutionary computer vision. He has published over 100 conference and journal papers.
1 Introduction

This paper describes the construction of an artwork using a software system for evolutionary art (Barile et al., 2009) based on genetic programming (Koza, 1992; Langdon et al., 2008). The artwork, ‘Can’t see the forest’, consists of a grid of animated drawings of trees created by using a software program we call ‘The Shroud’ (after the Shroud of Turin), for its capacity to produce a sort of ghost-like rendering or trace of a target image. We explore this evolutionary art process by describing the created work and the artistic and technical considerations that we encountered in its production. The goal of this collaboration between artist and computer scientist was ultimately to produce an engaging animation suitable for exhibiting in an art gallery.

The Shroud software produces visually engaging animated movies drawing grey level straight-line strokes at various angles and of finite lengths that give a variety of likenesses to a photographic or drawn target image. Figure 1 shows a sequence from an animated movie showing the target with a series of frames of the subject gradually emerging.1 (Note that ‘animation’ here refers to the emergence of the target tree, not that the target tree is moving or ‘animated’ in some way.) The movies are of two main kinds:

1 start with a canvas on which a large number of lines are drawn and keep redrawing the canvas until a recognisable subject emerges (see Figure 1)

2 start with a blank canvas and keep adding lines until a recognisable subject emerges (see Figure 2).

Figure 1 Sequence from a run of the evolutionary algorithm
In both cases, the animations engage the viewer’s attention as the viewer waits to see what will emerge. The challenge for the artist is to use the programs to produce animations that are not too fast or too slow, that have interesting targets, that keep the viewer in suspense for long enough, but not too long and to assess to what level of clarity the target will be revealed.

Having artists working with computer scientists produce some interesting cultural and technical issues. For example, in genetic programming from a computer science perspective, the desire would generally be to find an optimal result quickly and utilising the fewest computational resources. However, because we are here concerned with generating perhaps unusual results based on subjective, aesthetic criteria, the focus is more on ways for the programmer to give the artist increased creative control, rather than just finding the best level of fitness to a target image as quickly as possible.

The finished artwork consists of a seemingly infinite loop of animated ‘drawings’, with no clear start, middle or end. Each panel converges more or less towards the target image with its own starting and ending points on the main loop. The duration of the individual loops also vary (being factors of the main loop). The overall effect is to produce a dynamic piece with a balance of resolved and semi-resolved renderings of the target images overtime. Figure 6 shows an example snapshot from the animation.

2 Related work

A wide variety of evolutionary approaches have been used to generate art works, e.g. genetic algorithms (Ciesielski et al., 2007; Collomosse, 2008) genetic programming (Dipaola and Gabora, 2009; Machado and Cardoso, 2002) and ant colony optimisation (Monmarche et al., 2008; Semet et al., 2004). A number of these art works have been on a botanical/flora/plant growth theme. In Karl Sims’s (1990) Panspermia, the first evolutionary artwork to have a major impact, a simulated botanical environment was built using a genetic model of growth. Sims later expanded this work with genetic images (Sims, 1991), in which viewers were able to interactively resequence genetic markers used in the production of evolving images. There has been a stream of work based on evolving artificial plants. This work is based on L-Systems – simple languages that allow rules to be generated that model plant growth. A survey of this work is presented in McCormack (2008).

Non-photorealistic rendering (NPR) is the process of producing an image that does not look like a photograph, but is in some other style such as water colour or pencil
sketch. A number of researchers have used evolutionary techniques to generate NPRs. Some of this work is reviewed in Lewis (2008). In early work by Baker and Seltzer (1994), a genetic algorithm representation and interactive fitness evaluation are used to evolve line drawings of faces. Chakraborty and Kang (2004) describe an approach to rendering a target image using a genetic algorithm representation and a very simple set of flat rectangular brush strokes. Collomosse (2008) describes an investigation into the generation of painterly renderings in which a genetic algorithm representation is used for representing brush strokes and associated parameter values. Semet et al. (2004) generate painterly and pencil sketch renderings using an ant colony model. Edge maps and different models of the depositing of pheremones are used to achieve the different artistic effects. The user changes the parameter settings during the course of the run to guide the evolution according to their desired aesthetic criteria. Neufeld et al. (2008) describe an approach to evolving artistic filters that produce NPRs of a target image. An empirical model of aesthetics is used in conjunction with genetic programming and multi-objective optimisation. This work uses automated fitness assessment rather than human judgement.

Most of the work in evolutionary art has tended to focus on evolving static images, the installation of McCormack (1994) being a notable exception. Evolutionary techniques have been used to generate animated sequences of the development of locomotion in artificial creatures (Chaumont, 2007). Hart (2007) has developed animations in which an evolved image, represented by a mathematical expression tree, is morphed into another evolved image by a genetic cross dissolve process. The cross dissolve process is computed as a sequence of crossover and mutation operations. In Ciesielski et al. (2007) and Wijesinghe et al. (2008), the authors describe the evolution of animations of photomosaic images. Frames of the animation are composed of small tiles and the subject of the animation gradually emerges from random tile placements.

Many contemporary artists are producing work (installations, painting, photography, etc.) that responds to issues of sustainability in the environment. Patricia Piccinini’s work ‘Signs of Life’ from the ‘Plasticology’ series consists of ‘50 monitors with computer-generated plants constantly swaying on the wind’ Piccinini (1991), inspired by the artist’s ongoing exploration of ‘bio-scientific practices of manipulation and alteration of living beings, of creating new worlds’ (Haraway, 2007). Lyndal Jones’s ‘Avoca Project’ responds to the impacts of climate change in a novel way through intervention in the environment and a series of site-inspired artworks (Jones, 2009). Our artwork was also inspired by the pressing problems evident today such as global warming, deforestation and habitat destruction.

Repetition is an important creative practice for artists, whether used in the process of working up ideas, (a painter might do many small studies of a subject before the final, full-size painting), or as an essential feature of the finished work. Impressionist painter Claude Monet produced many studies of repetition of a theme over his long career: of haystacks, cathedrals and of the water lilies in his garden at Giverny, ‘in order to display minute discriminations of perception, the shift of light and colour from hour to hour on a haystack, and how these could be recorded by the subtlety of eye and hand’ (Hughes, 1997, p.359).

Pop artist Andy Warhol also used repetition extensively in his many series of popular culture and iconic figures. However, his silk screen print grids are not about recording changes in light but his repetitions instead ‘mimic the condition of mass advertising’ (Hughes, 1997, p.359) speaking ‘eloquently about the condition of image overload in a media-saturated culture’ (Hughes, 1997, p.540). Warhol produced variations on a single
image, repeating ‘these images until repetition is magnified into a theme of variance and invariance, and of the success and failures of identicalness’ (Wilson, 1968).

3 Artistic inspiration

The artwork interrogates concepts of environmental sustainability through creating a digital dying ‘forest’ (through repetition in a grid format, which suggests a continuation beyond the frame) of evolving and devolving animated movies of dead eucalypt trees. The idea of using genetic programming to create this work seemed an appropriate way to explore the effects of deforestation and global warming on our fragile and continually evolving environment in Australia. It also seemed apt that the programs were represented as tree structures (Figure 4), while the targets are also literally trees. The eucalypts photographed are naturally hardy and long-lived, however many have succumbed to drought in the last few years and can be seen in paddocks on farms and in forests, weathered silver and ghost-like.

The artist lived in Tasmania during the heyday in the 1970s of clear felling for wood-chipping in the old growth forests. The artist had been dismayed to see wind-rows of the leftover trees bulldozed into heaps for eventual burning. In the Shroud program, when the tree size is small there are very few lines drawn and the aesthetic effect is very like a game played by the artist when she was a child called ‘Pick Up Sticks’. It was played by letting the sticks fall like spaghetti into a saucepan – holding them upright and then letting them fall into a heap. The trick was to pick up as many sticks as you could without making the others move. The simple, straight lines drawn in the Shroud software reminded the artist of this childhood pastime, only then for her consciousness to shift back to the tragedy of the piles of logs in the Tasmanian forests. In ‘Pick Up Sticks’, the black sticks earned more points. In the wind-rows of bulldozed trees, they would all be burnt to black.

Artistic possibilities were suggested during the process of software testing. Normally, an artist constructing a drawing would use a rough sketch as a beginning, which would then usually be worked up into a more finished piece. However, as a result of doing a large series of runs and then playing them alongside each other on the computer screen as they evolved differently and at different rates, the artist was inspired by the creative and narrative possibilities inherent in combining both the ostensible ‘failures’ and the ‘successes’. Thus the software experimentation drove the idea of using a grid of multiple animations for the final artwork with some not fully resolving.

The artist was not interested in simply reproducing the target images in another form – animated drawings. She wanted to extend the creative possibilities by utilising the well-established art practice of repetition. The software was ideal for creating a digital dying ‘forest’ as it allowed for almost limitless versions of the target. However, the concept of regeneration is also present in the process, as the animations on a loop continually evolve towards the tree targets and then dissolve into random chaos, only to be reborn again.

4 Shroud algorithm

Evolutionary algorithms are characterised by a population of potential solutions that improve over the course of the evolutionary process. In our case, an individual is a
program that draws an image on a canvas and the measure of improvement, or fitness, is resemblance to a target image.

4.1 Representation

We have chosen to use a sequence of pencil lines to generate a NPR of a target image. The lines we use are of constant thickness and length. The thickness is one pixel and the length is computed from the width and height of the target image. Each line is rendered by a function: \( \text{Draw}(x, y, \text{angle}, \text{grey}) \). The \( x \) and \( y \) parameters specify the starting point of the line, the \( \text{angle} \) parameter specifies the direction and the \( \text{grey} \) parameter gives the darkness. All of these parameters are floating point values scaled to the range \([0, 1]\). An example program is shown in text form in Figure 3 and in tree form (not too be confused with the tree images that are the targets of the animations) in Figure 4. Executing this program will result in four lines being drawn on the canvas. Programs which generate the images shown in Figure 1 have around 800 lines. In drawing a line, two situations require special attention:

1. the line specified in the draw function will extend beyond the bounds of the image
2. a pixel is to be drawn on top of a pixel which was already drawn in a previous stroke.

In the first case, the line is truncated. In the second case, the default action is to average (or blend) the two pixel values. By default, each program starts with a blank canvas.

Figure 3  Text representation of an evolved program that draws four lines

\[
\text{prog2} \\
\quad \text{prog2} \\
\quad \quad \text{Draw}(0.465287, 0.114217, 0.600142, 0.769688) \\
\quad \quad \text{Draw}(0.243783, 0.109234, 0.374054, 0.407622) \\
\quad \text{prog2} \\
\quad \quad \text{Draw}(0.680495, 0.244189, 0.135088, 0.234363) \\
\quad \quad \text{Draw}(0.35214, 0.331753, 0.009837, 0.125121) \\
\]

Figure 4  Tree representation of an evolved program that draws four lines
4.2 Fitness

The fitness of an individual is the similarity of the rendered image to the target image. Fitter individuals generate images that are more similar to the target image. Our measure of fitness is the sum of pixel differences between the evolved image and the target:

$$\sum_{i=1}^{\text{width}} \sum_{j=1}^{\text{Height}} |\text{evolved}_\text{image}(i, j) - \text{target}(i, j)|$$
4.3 The evolutionary search

To generate an image that resembles the target, we need to find a program that contains an appropriate number of Draw commands with appropriate parameter values. Given the representation described above, there is a virtually infinite set of programs, most of which will generate images that look nothing like the target. We use an evolutionary search based on genetic programming to find good programs. The evolutionary process proceeds in the following manner:

1. Generate an initial population of random programs. These programs will contain some random number of Draw commands with random floating point numbers as the parameters.
2. Get the fitness of each individual by executing the program to generate an image as described in Section 4.1 and computing the fitness as described in Section 4.2.
3. Compute the next generation of individuals by applying the genetic operators of elitism, selection, crossover and mutation:
   a. Copy the best individual from the current generation to the next generation unchanged. This is elitism and ensures that the best individual in the next generation will be at least as good as the best individual in the current generation.
   b. Select two parents at random, but giving a higher probability of being selected to fitter individuals. Randomly select and swap subtrees (Figure 4) of the two parents to generate two children and place them in the new generation. This is the crossover operator and is analogous to the exchange of genetic material between two biological parents in mating.
   c. Select a parent at random, but giving a higher probability of being selected to fitter individuals. Randomly select a subtree, and replace it with a new, randomly generated subtree. This is the mutation operator and is analogous to the random change in genetic material that occurs in biological mating.
4. If maximum number of generations is exceeded or the fitness goal is achieved then stop, or else go to step 2.

The evolutionary search is a process similar to Darwinian evolution through natural selection. Programs with higher fitness have a greater probability of being selected for mating. Parts of each parent go into the offspring. The theory is that good building blocks (the parts of programs that render brush strokes) of programs with higher fitness will be propagated into a new generation, while building blocks that do not promote high fitness will disappear from the ‘gene pool’. The evolutionary process is terminated when a pre-specified number of generations have been completed or a pre-specified level of similarity to the target image is reached. Whenever a program is found that has a better fitness than the best program in previous generations, a frame of the final animation is generated. At the end of the evolutionary run, these frames are assembled into an animated movie. Figure 5 shows the change of fitness in a typical evolutionary run. It can be seen that the fitness monotonically improves as the number of generations increase. Interestingly, we have found that the optimal number of individuals in a population is four. This is very unusual in genetic programming applications where larger population
sizes tend to give better performance. The number of generations needed depends on the target image, choice of evolutionary parameter values and the desired similarity to the target. This can range from several hundred to several hundred thousand. More details of explorations of genetic representations for this problem can be found in Barile et al. (2009).

There are a number of parameters which control the evolutionary and drawing processes. These are shown in Table 1.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Parameters of the evolutionary and drawing processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program tree size</td>
<td>The maximum number of strokes that can be drawn. Larger programs draw more strokes.</td>
</tr>
<tr>
<td>Number of generations</td>
<td>The number of generations to execute before stopping.</td>
</tr>
<tr>
<td>Fitness target</td>
<td>Stop when the fitness reaches this value.</td>
</tr>
<tr>
<td>Write Nth best</td>
<td>Rather than rendering every best individual discovered as a frame, only write every Nth.</td>
</tr>
<tr>
<td>Fitness write threshold</td>
<td>Only write a frame if the fitness has improved this amount since the last frame written.</td>
</tr>
<tr>
<td>Accelerated convergence</td>
<td>In the basic program, a pixel on the canvas is always changed based on values in a draw function. If this parameter is set a pixel is only changed if the new value would be closer to the target value.</td>
</tr>
<tr>
<td>Incremental rendering</td>
<td>Rather than starting with a blank canvas, the rendering starts with the best canvas from the previous generation.</td>
</tr>
</tbody>
</table>

5 Producing artwork with ‘Shroud’

The artist exercises creative control by choice and pre-processing of targets and by choosing values for the evolutionary and drawing parameters described in Table 1. Some combinations of parameters lead to unusual and unexpected effects.

5.1 Selection and preparation of suitable target images

The trees were photographed by the artist, who is also a photographer, so as to include minimum background distraction so the tree stands out clearly against a light-coloured sky. The artist tried, where possible, to keep the scale and the horizon line reasonably constant for continuity between individual animations within the final piece. The target images chosen needed enough contrast and strong outlines so that even in animations using small programs, we would be able to make out the beginnings of a ‘drawing’ of a tree, even though very few lines are drawn.

5.2 Choices for the artist

1 How close to the target should the final image be? In some situations, photorealism is desired. In other situations, a fuzzy, suggestive abstraction is desired. The level of photorealism is controlled by a number of parameters acting in concert. Large values
for max generations and program tree size and turning on incremental rendering will give more photorealism. Beyond a certain number of generations, there is often minimal improvement. However, too few generations will often not produce anything resembling the target, particularly with smaller tree sizes.

2 At what rate should the target emerge? Usually, the artist’s goal is to have the target emerge at a rate that will keep the viewer’s interest. If the target emerges too fast there is no mystery or anticipation. If the target emerges too slow the viewer loses interest. The rate at which the target emerges is primarily controlled by the parameters write Nth best and fitness write threshold. Smaller values of these parameters will give longer animations or animation segments and more frames to be viewed before the target is revealed. Setting accelerated convergence and incremental rendering will generally decrease the time before the target is revealed. Setting a large program tree size and turning on accelerated convergence can result in the target being revealed in the very first frame.

3 Should the target be built up from a blank canvas by adding more and more lines or should it emerge slowly from a large number of randomly drawn lines? A small program tree size and incremental rendering will result in the former effect, as in Figure 2, while a large tree size without incremental rendering will give the latter effect, as in Figure 1.

4 How much energy should be in the animation? Some animations exhibit a considerable energy in that lines are continually being drawn and redrawn. In other animations, the drawing of the lines is not as obvious and the target slowly emerges without such vigorous line drawing on the canvas. Generally, small program sizes and incremental rendering turned off give the former effect while larger program sizes and incremental rendering turned on give the latter effect.

The individual images in Figure 6 are frames from animations that have been created with a variety of choices for the parameters described in this section. Combinations of different choices can give quite a wide range of different effects, some desirable and others not. For example, large tree sizes and incremental rendering can give a too obvious resolving of the target in the first generation. Conversely, turning off accelerated convergence and incremental rendering can produce many thousands of generations and still not reveal the target.

6 Reflection on challenges and opportunities

The artist’s main practice is still photography. Working with series of images to form a narrative is common practice in photography. However, the opportunity to work with subtle repetitions of an image and then animating the results into video format for presentation was creatively stimulating.

A challenge for the programmer is that the evaluative process is done on subjective and aesthetic (rather than efficiency) criteria. It can be challenging for a programmer to evaluate success without using objective and clear criteria. Ideally, in evolutionary programming, the animation would evolve to closely resemble the target image in as few generations as possible. But for the artist planning, a final artwork using many different animations in a grid or on multiple screens, it might be more desirable to be able to utilise
An artist’s experience in using an evolutionary algorithm

An artist’s experience in using an evolutionary algorithm

the steps along the way in the artwork, as opposed to using only the final or ‘best’ result. An artist, for whom the process is often the most engaging part of the work, will often react creatively to accidents rather than having a pre-defined set of criteria for successful outcomes.

It was a challenge in the beginning for the artist (used to extensively tested, more visually oriented and intuitive interfaces in software) to get used to the language of the programmer and be able to develop enough familiarity with the software to get the most out of it and to describe to the programmer what effects she wanted and why. The artist would then often forgo using the newer, improved, more efficient versions of the software and use the older, slower ones for artistic effect or only use the quicker versions sparingly. This could potentially have presented difficulties if communication between artist and programmer had not been strong. The programmer saw himself in this process as a mediator between the artist and the machine.

In Greenfield (2006), Greenfield contrasts the activity of artists in collaboration with programmers against the activity of what he calls ‘programmerartists’ – programmers who produce computer-generated art without the collaboration of professional artists or designers. Another term that Greenfield uses to describe these types of programmers is ‘algorithmic artists’. In his own words, the programmer who contributed to this paper describes himself as an ‘algorist’.

In that sense the term ‘algorist’ refers to two states: on the one hand that programming computer-generated art is an attempt to produce art from static code and on the other hand that creating software that generates art in a manner that, to the best of one’s knowledge, has never been done quite the same way before is an act of artistry itself. Greenfield examines that dual aspect of the ‘programmerartist’ in Greenfield (2006) and concludes that ‘art by computer program is evidence of programmer creativity, not computer creativity’.

The programmer who contributed to the work detailed in this paper had to fulfil both the roles of ‘programmerartist’ and that of collaborator with an artist. The role of ‘programmerartist’ dominated early in the work, but diminished in importance as the artist was given control of the usage of the software. The programmer often saw himself as merely the provider of the tools, such as one who merely produces brushes and canvasses for an artist to use. The programmer was content to provide the tools and let the artist do creative things with them.

But this view had to be tempered by the fact that the artist was implicitly dependent on the programmer to provide a new set of tools. The artist was saying in a roundabout way, “Give me something that Photoshop can’t give me”. This was quite a challenge for the programmer. It was not a case of programming a ‘photoshop-killer’, but of providing a means for the artist to explore creative effects not previously experienced by the artist.

The process was one of discovery for both collaborators. The programmer faced the challenge of engineering a somewhat unpredictable evolutionary process whilst completely delegating the task of determining aesthetic appeal to the artist. Overall it was a satisfying experience for both collaborators, as one gained a new toolset to play with and the other had the satisfaction of seeing his software used to produce works that he could not even imagine.
7 Conclusions and future work

‘Can’t see the forest’ is a work that has been created by exploratory and systematic use of a software system based on evolutionary algorithms. It adds creatively to the conversation around issues of environmental sustainability. The feedback loop established between the artist and the programmer functioned to quickly solve technical problems and to provide artistic inspiration through experimenting with the different iterations of the software as development progressed. This iterative process was very satisfying and fruitful creatively.

A major frustration for the artist with the use of the Shroud software is that some of the runs take hours or even days. In future work, we plan to develop algorithms that will run much faster and give the artist much earlier feedback on the consequences of parameter choices. We also plan to give the artist more control over the length and thickness of the lines drawn, to work with coloured lines and images, and to look at the possibility of changing the parameters during the course of a run.

Acknowledgements

We thank our colleagues Dr. M. Berry, D. Keep and Dr. D. D'Souza for suggestions relating to this paper and RMIT’s Design Research Institute for seed funding.

References


Note

1 Animations of some of the panels are at http://evol-art.cs.rmit.edu.au/npr_theshroud/.