Tatsuo Unemi

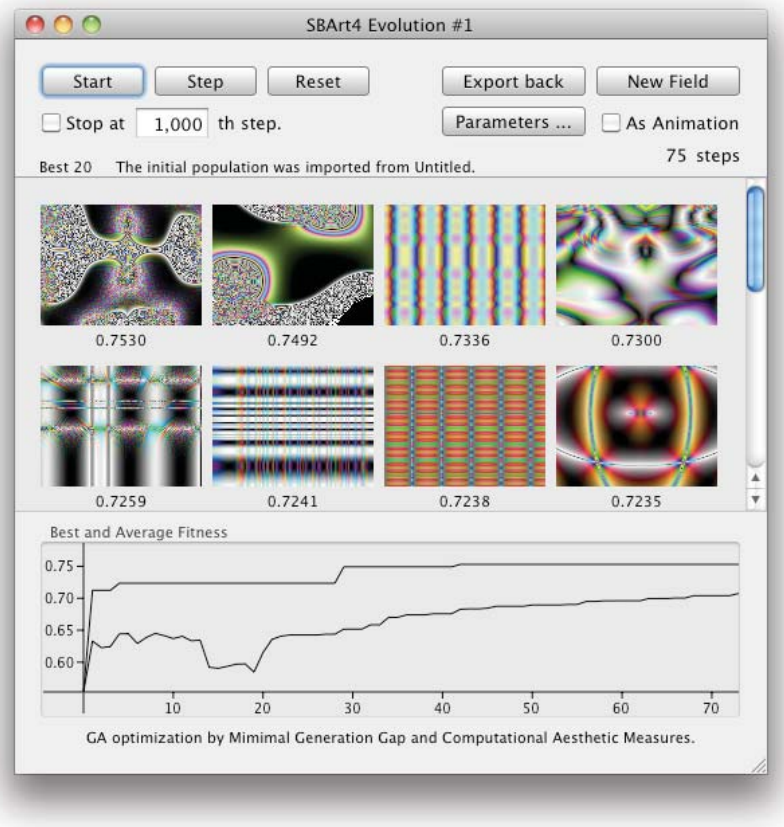
Paper: **SBArt4 as Automatic Art and Live Performance Tool**

Topic: Visual art

Authors:
Tatsuo Unemi
Soka University,
Department of
Information Systems
Science
Japan
www.soka.ac.jp

References:
[1] Tatsuo Unemi, "*A Breeding Tool for Abstract Animations and Its Applications*", 13th Generative Art Conference, 451-458, Milan, Italy, 2010
[2] www.intlab.soka.ac.jp/~unemi/sbart/4/

Abstract:
SBArt4 is originally a breeding tool to produce abstract images and animations [1]. This paper describes a newly implemented functionality of automated evolution using some types of computational aesthetic measures as fitness criteria. Measures include information complexity, global contrast factor, histogram of color distribution, and so on. In addition to these measures for a still image, a method to evaluate the motion was introduced. They are useful to discard boring pieces but do not always fit with human user's preference.
I introduced graphical user interface to allow the user to adjust the balance among different measures, and provided methods to let individuals migrate between a field for breeding and a population for evolution. It realized a wide range of intermediate production style between pure breeding and fully automated evolution.
Utilizing parallel processing, efficient algorithms, and synchronized sound effects, a new generative style of both live performance and fully automatic art became possible to be realized.



A sample shot of monitor window of evolutionary process.

Contact:
unemi@t.soka.ac.jp

Keywords:
Evolutionary art, abstract animation, aesthetic measures

SBArt4 as Automatic Art and Live Performance Tool

Prof. T. Unemi, BEng, MEng, DEng.
Department of Information Systems Science, Soka University, Hachioji, Japan
www.intlab.soka.ac.jp/~unemi/
e-mail:unemi@iss.soka.ac.jp

Premise

SBART [1] is originally a tool to produce abstract images and animations by means of interactive evolutionary computation [2]. Through a number of revisions since 1993 following the innovations of both hardware and software, it was equipped with capability for real-time breeding of animations [3]. This paper describes a newly implemented functionality of automated evolution using some types of computational aesthetic measures [4] as fitness criteria. Measures include information complexity, global contrast factor, histogram of colour distribution, and so on. In addition to these measures for a still image, a method to evaluate the motion was introduced. They are useful to discard boring pieces but do not always fit with human user's preference.

A graphical user interface was designed to allow the user to adjust the balance among different measures, and provided methods to let individuals migrate between a field for breeding and a population for evolution. It realized a wide range of intermediate production style between pure breeding and fully automated evolution. Utilizing parallel processing, efficient algorithms, and synchronized sound effects, a new generative style of both live performance and fully automatic art became possible to be realized.

1. Breeding Real-time Animations – Overview

SBART is one of derivative systems from K. Sims's Artificial Evolution in 1991 [5] that uses functional expression as a genotype to determine the colour of each pixel in a rectangular area. The phenotype is the image constructed by these pixels that the user evaluates and selects as a parent to produce individuals for the next generation.

The first release of SBART was in 1994 on UNIX workstations utilizing Motif GUI gadgets and X window system, and has been extended and migrated to MacOS 8, MacOS X on Power PC, and then MacOS X on Intel CPU. A number of features have been examined and added through these 17 years. One of the unique features in SBART is that it uses a vector of three elements as a value in any point in calculation. Some of the non-terminal symbols for functional expression on genotype calculate the value of each element in vectors independently, but the others mix the values to reconstruct the result value. The terminal symbols are also vectors. A variable is a permutation of three scalar variables, x, y and t, that express the coordinate of the pixel in 2D space and time. The other unique features include multi-field user interface, some alternative drawing modes of deformation and discoloration for external image data, synchronized sound effects for animation, and so on.

The most recent innovation is automated evolution combined with real time breeding of animations. Thanks to the recent progress of GPU technology, it became possible to draw 15-30 images within one second. To draw an image in a rectangular area, it requires computational time proportional to the number of pixels in the image and the

number of elemental functions in the genotype. Because the genotype is the object for mutation and crossover, it is represented in a data structure in the memory. It is not technically impossible but difficult to develop a mechanism of so called on-line compiler for each functional expression in genotype under ordinary programming environment of C, C++, and Objective-C languages.

To take an advantage of GPU that originally developed to accelerate calculation for 3D polygon graphics, the shading language is suitable for our purpose. A graphics library OpenGL accompanying with a GPU vender's extension includes the compiler of a programming language GLSL that has similar syntax of C language, and it compiles the program text in the memory into the machine code for GPU. This means that any application software can utilize the power of GPU by only prepare its aiming procedure in GLSL to rely on the compiler. The newest version of SBART compiles the genotype into Core Image Kernel Language, a subset of GLSL provided by Apple as a part of Core Image framework. Program fragments run on the processing elements in GPU in parallel for each pixel. The number of pixels it calculates at same time is depends on the capacity of GPU, but all of Apple's personal computers released in these three years have capability to draw smooth animations in real time.

2. Automated Evolution

The criteria for artistic production should fundamentally be under subjective judgement of each individual person, but it's also the fact that there are sharable measures among people for interestingness of any pieces. Some of the challenges of computational aesthetic measures are to find such criteria to evaluate and select better images from digitized pictures and paintings.

The aesthetic evaluation done by human is fundamentally driven by the cognitive system of individual person, of which functionality is strongly depending on the personal experiences under the cultural background. To implement such type of functions in the computer, it is necessary to use a type of learning mechanisms to adapt to a variation among persons and changes through the time in an individual person. Some researchers are trying to embed such adaptability utilizing algorithms of artificial neural networks, for example in [5]. However, we avoid such an adaptive mechanism but consider only common criteria that a majority of people will agree to use to omit some types of uninteresting images as an assistant mechanism for efficient breeding. Therefore, a style of paintings of abstract art, such as a painting with a single colour in the whole area of the canvas, is out of scope from this approach, because it usually requires knowledge of the historical and cultural context to interpret the piece appropriately.

The following subsections describe three types of measures for spatial arrangement and two types for colour variation. To unify these measures in different dimensions for a final fitness value, a normalization method is introduced as described in the following subsection. After description of the measure for motion in animation, the method to alternate generations in evolutionary computation is described in the final subsection.

2.1 Information theoretic complexity

Commonly used measure is information theoretic complexity. Intuitively saying, simple uniform pattern, such as a solid colour, should be eliminated from the candidates for interesting images. Theoretically, Kolmogorov complexity in

information theory is the ideal concept to measure how meaningful information the data contain. However, the computation of this measure cannot always correctly be completed in a feasible time and space. For example, a pattern of pseudo random dots is usually describable by a simple algorithm and a few parameters, but it is difficult to find the parameter values from the data even if the algorithm is given. The alternative method to approximate this measure is to employ an algorithm of data compression, such as JPEG compression for two-dimensional image. P. Machado *et al* examined some different types of methods for aesthetic measure from the complexity [6], and J. Rigau *et al* made deeper consideration on the complexity as aesthetic measures [7]. In this system, only one simple method has been implemented, that is, to measure a distance between the compression ratio of the given ideal value and the real value measured from the object image, as same as E. den Heijer *et al* examined [8]. It was implemented utilizing an API of JPEG compression embedded in MacOS X Cocoa framework with the image quality parameter as 0.67. Because the ideal value of compression ratio should be a subject to alternate due to the user's preference, this system allows the user to adjust it using a graphical user interface (GUI) described later.

2.2 Global contrast factor

E. den Heijer *et al* employed Global Contrast Factor (GCF) [9] as an alternative measure for interesting pattern [8], of which algorithm was originally designed to evaluate *contrast* as nearer with human's intuitive measure as possible. Contrast means the difference among brightness in a single image, but it is more than a simple statistical measure of variance among brightness values over the pixels. It should be called high contrast if the image is organized only black and white pixels without any intermediate gray ones, but it should be called low contrast if the allocation of black and white for each pixel was randomly determined. K. Matkovic *et al* proposed the algorithm to calculate a weighted summation among average differences over all of pairs of neighbouring pixels for different resolutions of a single image [9]. Their original method uses a greyscale image of 1024 × 768 pixels as the original one, and reduces the resolution in half by half in seven steps until it reaches the smallest size of 4 × 3 pixels. The weight values were statistically induced through the psychological experiments with human subjects.

To expand it to be applicable to a colour image, the difference between brightness is changed into the Euclidean distance between colour values in RGB colour space. Three component values are scaled in 2:3:1 for red, green and blue to adapt to the characteristics of human eyes.

In the current implementation in SBART, it starts the calculation from the half size of the original one, that is, 512 × 384, because it needs to keep the computation time short enough for interactive use combining with simulated breeding. This point will be revised by developing more efficient algorithm using GPU and improvement of hardware performance.

2.3 One-dimensional brightness distributions

The above two measures are concerning the placement of colours in an image, but they do not include any consideration on two dimensional distribution, that is, one dimensional patterns, an image of parallel stripes, are also given a chance to gain high aesthetic score. Two dimensional Fourier analysis should be useful to evaluate such a pattern expansion over the image, but it consumes long computing time of O(N log N) for each frequency. Another easy method to detect a pattern of parallel stripes is to compare the variances of brightness distribution among rows and

columns. If the pattern is horizontal stripes, the variance among rows is large, but the variance among columns is zero.

The algorithm implemented here is constructed by following three steps.

1. It calculates the distances between distributions of brightness in the ideal one and measured one for each angle from 0 to 90 degrees stepping by 15.
2. It transforms each result value to adjust zero to 1 and furthest to 0.
3. Then, it takes a geometric mean among the values.

Measurement for a variation of different angles is helpful to detect the orthogonal parallel stripes. The ideal distribution is extracted based on a statistical analysis over 500 snapshot photos similarly to the case of colour histogram described in the next subsection.

2.4 Color histogram

In addition to the shape, some statistics on colours is also important to index the characteristics of an image. The frequency distribution of brightness was examined in a previous work by E. den Heijer *et al* using the distance between the ordered frequency distribution and an ideal distribution based on Benford's law [8].Benford's law is that the appearance frequency of digit d at the highest column in prime numbers follows the value of $\log_{10}(1+1/d)$ [10]. The algorithm E. den Heijer *et al* developed counts pixels for each span of nine grades of greyscale from black to white over all of pixels in a single image, sorts the frequencies calculated from the counted numbers in descending order, then measures the distance from the distribution of Benford's law. Another distribution found from frequency of occurrence for each word in texts is Zipf's law that the frequency is proportional to the inverse number of the rank in descending order, that is, the second frequent word appears a half times of the first one, the third frequent word appears one third of the first one, and so on [11]. Both of these distributions can be found in a number of natural phenomena.

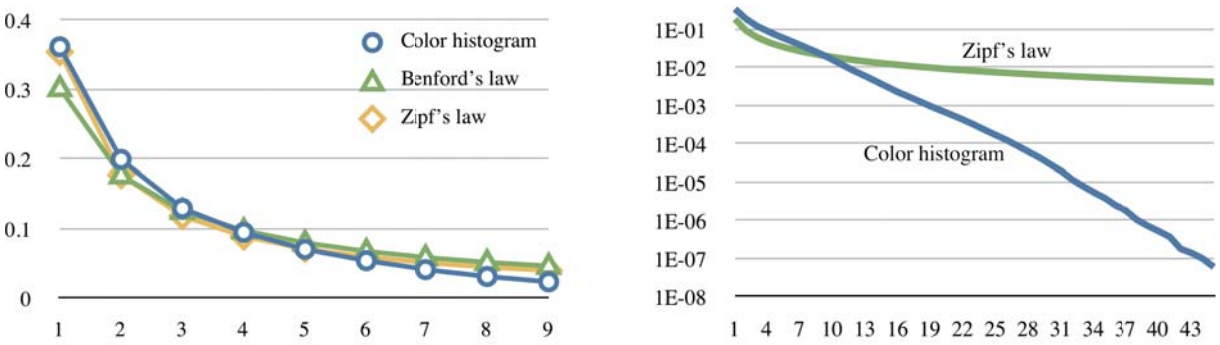


Figure 1. Comparison among distributions of average colour histogram extracted from 500 snap photos, Benford's law, and Zipf's law.

To extend the method applicable to colours, we divide the colour space into 6 × 9 × 3 for each component of red, green and blue, 162 clusters in total. Instead of using the above two well-known distributions, we investigated the average distribution among snapshot photos of nature, urban sceneries and portraits. The result using 500 photos showed almost following Zipf's law in the top nine colours, but the rest of low frequencies figured an exponential distribution as shown in Figure 1.

Therefore, as the ideal distribution, we embedded a sequence of frequencies in top 45 colours revealed from this investigation. The distance is calculated by taking the summation of the absolute differences between the ideal frequency and the calculated frequency for each rank in the same way of [8].

2.5 Favorable distribution of saturation

There is no universal aesthetic criterion on the tone of colours. However, the user, an image breeder, sometimes has preference of monotone or colourful. Here we introduce a user interface that allows the user to indicate the ideal values of average and standard deviation on saturation values over all pixels. A gray image becomes preferable if both the average and the standard deviation are indicated low, and a colourful image becomes preferable if the average is high.

When we define the range of saturation values within a fixed span, such as [0,1], the possible value of standard deviation is limited within a range depending on the average value. The standard deviation takes the maximum value when the sample values are restricted in the edge values of the range, that is, 0 and 1. In this case, the average μ and the maximum standard deviation $\sigma_{\max}$ are:

$$\mu = P_1 \tag{1}$$

$$\sigma_{\max} = \sqrt{P_0 P_1} \tag{2}$$

where P_0 and P_1 are frequencies of saturation values of 0 and 1 respectively, that is $P_0 + P_1 = 1$. The detail derivation of equation 2 is in Appendix. For the user's convenience, the user is allowed to input the value within [0, 1] as the standard deviation, and the system multiplies it by $\sigma_{\max}$ of equation 2 for actual ideal value. The measure is the two dimensional Euclidean distance between the ideal values and the actual values extracted from the image.

2.6 Normalization and unified measure

It is necessary to develop a kind of normalization for the five measures described in the above subsections, because they are in different dimensions for each that should not be compared as they are. We transform each measure in the range of [0, 1] so that 0 indicates the worst and 1 indicates the best, and map it to the normalized measure by two stages so that all of processed values form distributions of similar shapes. The first stage is a gamma correction $f_1(x) = x^{\gamma}$ so that the average value is equal to the median value, that is, $\gamma = \log_m \bar{x}$ where m is the median and $\bar{x}$ is the average value of x. The second stage is a linear transformation so that the average and the standard deviation are adjusted to 0.5 and 0.2 respectively. The equation is $f_2(x) = (0.2/\sigma_2)(x - m) + 0.5$ where σ_2 is the standard deviation among the transformed values by f_1. Combining these two stages, the transformation function f is defined as:

$$f(x) = f_2(f_1(x)) = (0.2/\sigma_2)(x^{\log_m \bar{x}} - m) + 0.5 \tag{3}$$

If the final value is out of the range [0,1], it is revised to the nearest boundary 0 or 1. To determine the coefficients in the transforming functions, we examined 1,000 images drawn with randomly generated genotypes. Figure 2 shows the distribution of each measure and its normalized version.

The final aesthetic measure for a still image is calculated as the weighted geometric mean among these measures, where the weights are adjustable by the user using a GUI in figure 3. It includes sliders for each measure to allow the user to operate any of them in anytime. Once one of these values is manually changed, the others are automatically adjusted to keep the summation of the weights to be 1 and to keep the ratio among the weights being adjusted if possible.

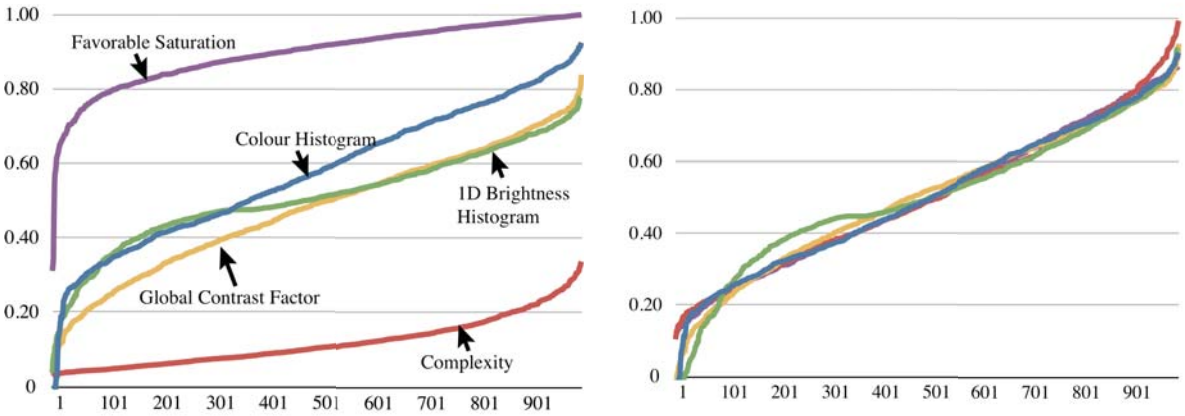


Figure 2. Distributions of aesthetic measures for 1,000 images produced from random genotypes. Each measure is sorted in ascending order.

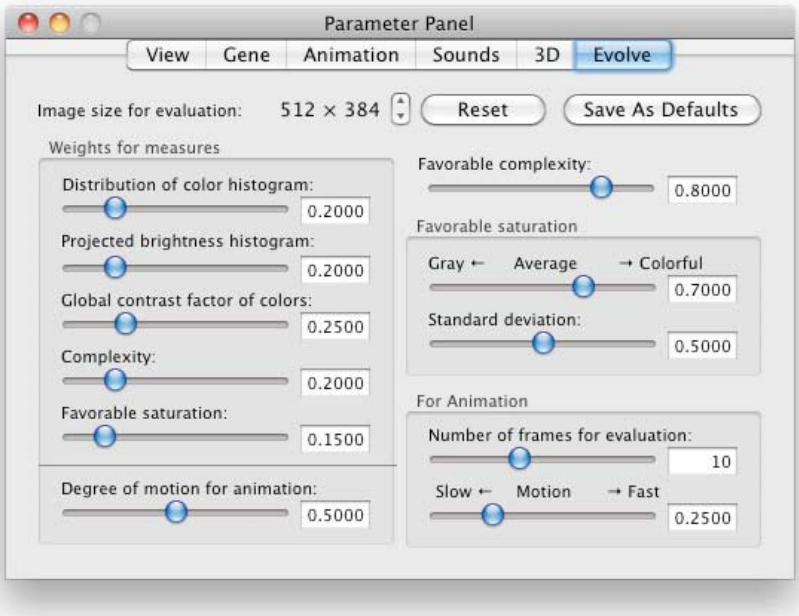


Figure 3. GUI for parameter settings of automated evolution.

2.7 Aesthetic measure for animation

One of the unique features of SBArt4 released in 2010 is that the user is allowed to breed not only still images but also animations in real time [3]. It used be difficult because the required computation time to draw one frame was much longer than the usual frame rate for smooth animation, but the recent improvement of GPU made it possible to realize it. We tried to implement a measure of favourable degree of movement in animation to be combined with the measures for still image. Ideally saying, this type of measure should be constructed with total evaluation over whole

duration of the animation, but it seems very difficult because of too much computation power needed. As the first step of minimal functionality for this measure, we implemented the algorithm to calculate the average difference between consecutive frames among samples picked up from the whole sequence of frame images. It is theoretically possible to breed an animation of arbitrary duration, but to keep the evaluation by the user easy, SBArt4 treats relatively short animation, four seconds in default. It is a usual dilemma between a quick response and precision. To keep a smooth interaction between automated evolution and breeding, the current implementation uses ten samples in default to estimate the average motion by calculating the distance between colour values in pixels of same position in a sampled frame and the next frame. Because the degree of motion should be a subject to be adjusted according to the preference of the user, the system is providing a slider to allow the user to set up the value in anytime, that is located at bottom right in GUI shown in Figure 3.

The measure for still image is also applied to each frame image of ten samples. The total evaluation is calculated as the weighted geometric mean between the average measure of still images and the average degree of motion in sampled frames. The weight is also adjustable by the user using a slider at left bottom of the GUI.

2.8 Method of generation alternation

From the view point of the main objective of this system, the fitness value based on the measures described above is just a hint for users expected helpful for the user to find better candidates efficiently. We designed it not as an automatic (or artificial) artist but as a support system (or assistant) for human artists and designers. During the usage along with this principal, the system should allow the user to interrupt the evolutionary process to revise the parameters and the population in anytime. In a case with such a requirement of interruption, the generational change should be designed in a style of small-grained alternation to keep the computation time between generations short enough for human-machine interaction. Among the several methods available for this style of genetic algorithms, we chose Minimal Generation Gap (MGG) model [12] because of the smallest grain size and the most effective exploration in a large search space. In each step of generation alternation, the algorithm randomly chooses two individuals from the population as parents to organize a family by producing two children by crossover and mutation. Then it evaluates each members of the family if necessary. It selects the best individual among the family and randomly selects another individual from the family to restore the selected two individuals into the population. The other two individuals in the family are discarded. Random selections of parents and the second survivor are effective to keep wide diversity in the population, and the best selection has the same effect of the elitist strategy that guarantees the best individual discovered in the process always remains in the population.

We designed a GUI shown in Figure 4 to make it easy for the user to monitor the evolutionary process. The middle part of the window displays the best 20 individuals discovered in the process. The fitness value for each individual is also shown under the image. The bottom part shows a live animation of the trend graph on fitness values of both the best one and the average among the best 20. Each measure of both still image and animation is observable by a tooltip balloon attached to each view of the best 20 individuals that pops up when the mouse cursor stays some seconds on the view.

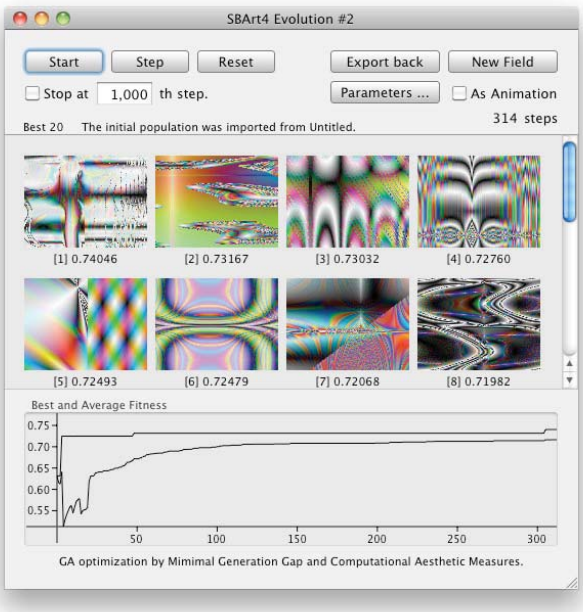


Figure 4. GUI for automated evolution.

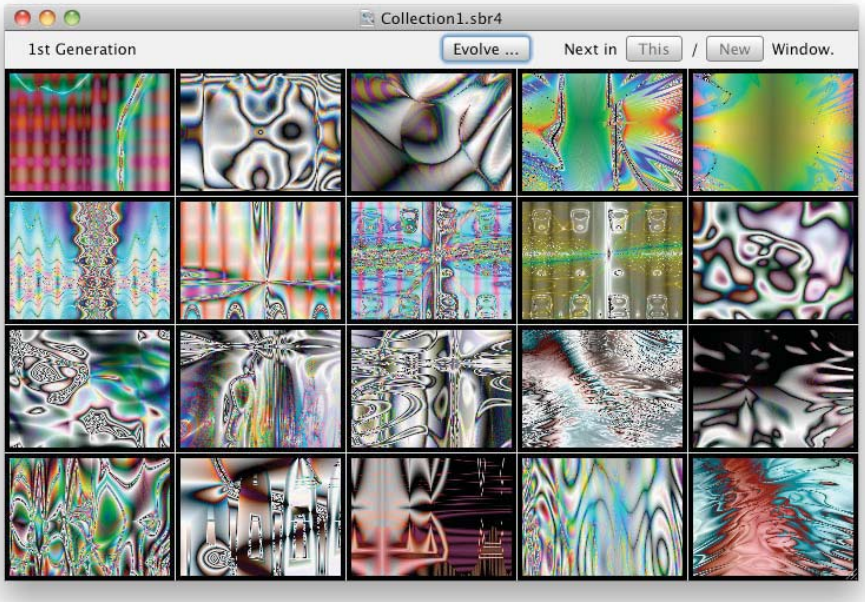


Figure 5. GUI for simulated breeding.

3. Combination of breeding and evolution

We designed a GUI for easy operation by the user to exchange the individual genotypes between the population in automated evolution and the field for simulated breeding. Figure 5 shows a sample field window of simulated breeding in SBArt4. New 20 individuals are organized as children, when the user selects his/her favorite phenotypes displayed in the window as the parents and clicks one of the buttons at top right corner of the window. A new button labeled “Evolve . . . ” was added by which the user open a new evolution window to start a new process of automated evolution as shown in Figure 4. Because the population size for evolution is 80 in the

current implementation, the initial population is organized with 20 individuals imported from the field window and the other 60 are generated via crossover and mutation from these 20 individuals by random selection. It is also possible to start an automated evolution from a pure random population from a menu item in the menu bar of the application independently from a field window.

Two buttons at top right of the evolution window are to export the best 20 individuals to a field window of the original field by the left button and a new field by the right button. In the same style implemented in SBART for multi-field user interface [13], any individual displayed as one of best 20 can migrate to arbitrary position of a field window by copy & paste and drag & drop operations. By these functions, the user can progress breeding from his/her favorite individuals produced though an automated evolution when he/she found them in the best 20 views.

The individuals in a field window are also able to migrate to a population of automated evolution already running by copy & paste and drag & drop. In this case, the destination place of migration is not in the best 20 individuals, but another individual in the population of which fitness value is not calculated yet or less than the migrant. When there is no appropriate place found in the population, the individual of the least fitness is replaced.

4. Breeder/Evolver and Displayer

It became possible to produce interesting abstract animations in real-time by the automated evolutionary process described above. This means it is theoretically possible to set up an installation of never-ending series of animations by showing sample individual in the population in turn. It must be a new style of automatic art, but we need to solve the following two issues to realize it.

The one is on a computing performance of the hardware. Because the processes both of evolution and of animation require much of computational power, a single portable computer is not powerful enough. It is feasible if a hi-end personal computer with eight cores and hi-performance GPU board is available, but portable ones are preferable to keep the installation easy in any place.

The other problem is to keep the animation played back smoothly even when the evolutionary process requests GPU resource to draw an image to evaluate its fitness value. In the case of breeding, this point is also a problem when the breeder wants to keep a bred animation played back during he/she is searching a next candidate for playback by breeding. The process of playback must have higher priority than both evolution and breeding if they share any of computing resources such as CPU, GPU, and memory.

To solve these issues, we employ two computers connected by a local area network, and developed two new application software. The one named *controller* runs on the same machine that SBArt4 is running, and the other one named *player* runs on the other. The main role of *player* is to receive a text of code written in shading language to draw the animation on the hi-resolution screen. *Controller* is a type of communication-bridge between SBArt4 and *player*. It receives a code through the general pasteboard usually used for copy & paste between independent applications, and passes it to *player* by TCP/IP connection through the network. It has a GUI to control player remotely to change parameters for timing and sound effects.

For the performance of real time breeding, the human player breeds his/her favourite animation on SBArt4 and send the code by copy & paste operation to *controller*. The pasted code is automatically sent to player running on another machine. If we used a single machine to run both SBArt4 and player, an extra equipment of audio I/O is required because a personal computer usually has only one pair of stereo audio channels for both input and output though we need two separated audio output for breeding and playback.

For the automatic art, we developed software in AppleScript that supervise both SBArt4 and *controller*. It initiates an evolutionary process in SBArt4 starting with random parameter settings and picks up 10 individuals after 50 steps to copy their codes into *controller* in turn. Then, it demands SBArt4 to reconstruct the population by genetic combination among the individuals in the current population. Some of the individuals are replaced with new random genotype to keep diversity in the population. Evolutionary process restarts again and continues until playbacks for all of previous 10 individuals are completed. These processes are iterated arbitrary times to produce ever-changing stream of abstract animations. The duration of an animation for each individual is 20 seconds in a typical setting, in which the evolutionary process is allowed to continue 3 minute 20 seconds to produce the next 10 individuals for playback. It is long enough for small computer to execute more than 50 steps of generational changes in MGG model described in section 2.8.

5. Conclusion

New mechanism of automated evolution of abstract animations is introduced to SBART. Each of the produced pieces is not perfect for human critics but it is useful to find interesting animations. This functionality enabled to implement a new style of installation of automatic art that shows visitors ever-changing abstract animations for long time. To run this installation smoothly in portable machines, two new applications were developed to use two computers. These new development also enabled a live performance of real time breeding of animations on stage.

The automatic art have been exhibited twice in the Open Campus in Soka University. We received positive comments from visitors at both occasions. For the breeding performance, GA 2011 is the first occasion to perform in public. The author hopes the audience enjoy it and having another chance to exhibit the installation.

References

[1] T. Unemi, SBART 2.4: an IEC Tool for Creating 2D Images, Movies, and Collage, Leonardo, Vol. 35, No. 2, pp. 171, 189-191, MIT Press, 2002.

[2] H. Takagi, Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation, Proceedings of the IEEE, Vol. 89, No. 9, pp. 1275-1296, 2001.

[3] T. Unemi, SBArt4 - Breeding Abstract Animations in Real time, Proceedings of the Conference on Evolutionary Computation 2010, pp. 4004-4009, Barcelona, Spain, 2010.

[4] L. Neumann, M. Sbert, B. Gooch and W. Purgathofer (Eds.):. Computational Aesthetics 2005: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging, May 2005.

[5] K. Sims: Artificial Evolution for Computer Graphics, Computer Graphics, Vol. 25, pp. 319-328, 1991.

[6] P. Machado and A. Cardoso: All the Truth about NEvAr, Applied Intelligence, Vol. 16, pp. 101-118, 2002.

[7] J. Rigau and M. Feixas and M. Sbert: Informational Aesthetics Measures, IEEE Computer Graphics and Applications, Vol. 28, No. 2, pp. 24-34, 2008.

[8] E. den Heijer and A. E. Eiden: Using Aesthetic Measures to Evolve Art, Proceedings of the Conference on Evolutionary Computation 2010, pp. 4533-4540, Barcelona, Spain, 2010.

[9] K. Matkovic and L. Neumann and A. Neumann and T. Psik and W. Purgathofer: Global Contrast Factor - a New Approach to Image Contrast, Computational Aesthetics 2005, pp. 159-168, 2005.

[10] J.-M. Jolion: Images and Benford's Law, Journal of Mathematical Imaging and Vision, Vol. 14, No. 1, pp. 73-81, 2001.

[11] G. K. Zipf: Human behavior and the principle of least effort - an introduction to human ecology, Hafner Pub. Co., New York, 1949.

[12] H. Satoh and I. Ono and S. Kobayashi: A New Generation Alternation Model of Genetic Algorithms and Its Assessment, Journal of Japanese Society for Artificial Intelligence, Vol. 12, No. 5, pp. 734-744, 1997 (in Japanese).

[13] T. Unemi: A Design of Multi-Field User Interface for Simulated Breeding, in Proceedings of the third Asian Fuzzy Systems Symposium, Masan, Korea, pp. 489-494, 1998.

Appendix: Derivation of equation 2

From the definition of standard deviation,

$$\sigma_{\max}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \overline{x^2} - \bar{x}^2 .$$

Because x_i = 1 or 0 and the expected number of x_i of value 1 is $P_1 N$,

$$\overline{x^2} = P_1 \text{ and } \bar{x}^2 = P_1^2 .$$

Therefore,

$$\sigma_{\max}^2 = P_1 - P_1^2 = (1 - P_1)P_1 = P_0 P_1 .$$