Querying Across Time to Interactively Evolve Animations

Isabel Tweraser
Southwestern University
Georgetown, Texas
twerasei@southwestern.edu

Lauren E. Gillespie
Southwestern University
Georgetown, Texas
gillespl@southwestern.edu

Jacob Schrum
Southwestern University
Georgetown, Texas
schrum2@southwestern.edu

ABSTRACT

Compositional Pattern Producing Networks (CPPNs) are a generative encoding that has been used to evolve a variety of novel artifacts, such as 2D images, 3D shapes, audio timbres, soft robots, and neural networks. This paper takes systems that generate static 2D images and 3D shapes with CPPNs and introduces a time input, allowing each CPPN to produce a different set of results for each slice of time. Displaying the results in sequence creates smooth animations that can be interactively evolved to suit users' personal aesthetic preferences. A human subject study involving 40 individuals was conducted to demonstrate that people find the dynamic animations more complex than static outputs, and find interactive evolution of animations more enjoyable than evolution of static outputs. The novel idea of indirectly generating artifacts as a function of time could also be useful in other domains.

CCS CONCEPTS

• **Applied computing** → **Media arts**; • **Computing methodologies** → **Neural networks**; **Generative and developmental approaches**;

KEYWORDS

Art, Animation, Interactive evolution, Indirect encoding

ACM Reference Format:
Isabel Tweraser, Lauren E. Gillespie, and Jacob Schrum. 2018. Querying Across Time to Interactively Evolve Animations. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205460

1 INTRODUCTION

Compositional Pattern Producing Networks (CPPNs [11]) are a generative encoding with a broad range of applications. These networks are a type of artificial neural network that have arbitrary topologies and whose activation functions are chosen from a variety of functions that help create interesting patterns. CPPNs are typically evolved to be queried across a coordinate frame, and indirectly encode patterns across that space. CPPNs have many applications: they have been used to generate 2D images [10], 3D shapes [2], audio timbres [5], soft robots [1], and neural networks [12]. Several of these examples allow users to evolve their own artistic artifacts using interactive evolution. Picbreeder [10] (see Section 2.1) is a prominent early example that allows an online community to generate 2D images. An interesting follow up is Endless Forms [2] (see Section 2.2) that instead evolves three-dimensional objects.

While Picbreeder and Endless Forms are innovative applications of CPPNs, they both produce static outputs. Endless Forms displays animations, but these animations are merely rotations of static shapes. In this paper, CPPNs are queried across time to create dynamic outputs. Specifically, AnimationBreeder and 3DAnimationBreeder are two new programs that expand on the original ideas of Picbreeder and Endless Forms by introducing a new input to CPPNs: time. By adding a time input to CPPNs and generating sequences of results over an interval of time rather than individual results, both 2D and 3D animations can be generated. This enhancement results in more complex and interesting dynamic results.

Users recognize the increase in complexity that the addition of a time input brings to these animating programs, in contrast to their static counterparts. A human subject study was conducted in which users interacted with one of the static programs and its associated animating program for several generations, and were then surveyed to assess their experience with each program. The study shows that users enjoyed the animated programs more than the static ones, and think that the animated programs produce more complex artifacts than their static counterparts. The study supports the notion that introducing time to interactively evolve animations is a novel new application of CPPNs that produces interesting new artifacts.

This paper proceeds by discussing previous work evolving artistic artifacts with CPPNs (Section 2), then delves into the technical details of how various interactive evolution programs are implemented (Section 3). In Section 4, the user interface is described. Then Section 5 presents the protocol and results of the human subject study used to asses the new approaches of this paper. A discussion of the results follows (Section 6) before the paper concludes (Section 7).

2 PREVIOUS WORK

This section describes previously implemented interactive evolution programs that are recreated and extended in this paper.

2.1 Picbreeder

Picbreeder[1] is an online interactive evolutionary art system that allows different Internet users to evolve results from simple random starting points, or from results created by others [10]. Picbreeder was an early example of the expressive power of CPPNs. The interface displays a series of images on buttons. Each image is generated by querying a CPPN at each pixel of the image. Users select images they like, and then go to the next generation to see similar, but

[1] http://picbreeder.org

GECCO '18, July 15–19, 2018, Kyoto, Japan
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205460

slightly different offspring images derived from their chosen images. Because the website is public and images can be saved, the process of evolving interesting images is collaborative: new users can take results produced by others and evolve them further. This system is referred to as collaborative interactive evolution.

The recreation of Picbreeder in this paper is similar to the original, but is not online and uses a different user interface. The details of differences between the two implementations are in Section 3.3, and a description of the new user interface is in Section 4.

2.2 Endless Forms

Endless Forms[2] is another application of CPPNs that extends the idea of Picbreeder to instead evolve three-dimensional objects [2]. Expanding the design space for CPPNs so that they can create objects in three dimensions allowed for new explorations of the capabilities of CPPNs. Endless Forms creates objects by querying CPPNs at every voxel in a three-dimensional space at a certain resolution, and then filling in a voxel if a CPPN output exceeds a certain numerical threshold. After the voxels have been filled in, the object is processed with the Marching Cubes algorithm [7], which smooths edges and corners to create a more cohesive, less blocky shape. These objects can then be evolved with an interface similar to that of Picbreeder.

Endless Forms is also an online collaborative evolution platform with a web-based interface. The objects rotate within their display buttons on the interface so that users can see them from multiple angles. The objects in Endless Forms are a single color, but an extension to Endless Forms using the same shape encoding added the ability to assign different colors to each individual voxel, creating multicolored 3D objects [6]. This extra color encoding is included in the reimplementation of Endless Forms used in this paper. These implementation details are discussed in Section 3.5.

3 METHODS

This section first describes the selective breeding algorithm used to interactively evolve artistic artifacts. Then CPPNs are described, followed by details on how they are used to evolve 2D images and animations, as well as 3D shapes and animations.

3.1 Selective Breeding

In order to evolve interesting artistic artifacts, a simple evolutionary algorithm similar to selective breeding is used. In each generation, the user sees the population of N options available for selection. The user can select M individuals as parents for the next generation, for $M < N$. After selections are made, the M parents are directly copied to the next generation, which is an example of pure elitist selection. Remaining slots in the next generation are filled with offspring until there are once again N members in the population.

Offspring are created by either randomly picking a single elite parent to create a mutated clone of, or by picking two random parents to crossover, before mutating the resulting child. The choice between these two options is made probabilistically based on the crossover rate. In the next generation, the selection process repeats with N new options. This selective breeding algorithm could apply to any representation, but CPPNs specifically are used in this paper.

[2]http://endlessforms.com

3.2 Compositional Pattern Producing Networks

CPPNs generate all artistic artifacts in this paper. A CPPN is a type of artificial neural network with an arbitrary topology containing a variety of activation functions with repetitive, symmetric, and asymmetric patterns. These patterns are hallmarks of natural organisms.

CPPNs are typically evolved with NeuroEvolution of Augmenting Topologies (NEAT [13]). NEAT evolves networks with arbitrary topologies using three possible mutations: weight mutation perturbs the weights of existing network connections, link mutation adds new connections between existing nodes, and node mutation splices new nodes along existing connections. Another key innovation of NEAT is topological crossover based on historical markers.

CPPN nodes can each have a different activation function, so every newly created node in a CPPN is assigned a randomly chosen activation function from a set of human-specified options. This set contains representatives of symmetric, periodic, and asymmetric functions in order to mimic patterns seen in nature. To allow better exploration of the space of possible functions, there is also a mutation operation that swaps the activation function of a random node with another randomly chosen activation function.

However, the main benefit of CPPNs is how they are applied: they are repeatedly queried across a coordinate frame in order to define an object within that frame as a function of its geometry. CPPN inputs are locations in the coordinate frame, and the resolution of the artifacts created depends on the proximity of adjacent query points. Therefore, CPPNs can generate artifacts at arbitrarily large resolutions, albeit at extra computational cost. Each program in this paper encodes objects using a different coordinate frame.

3.3 Evolving 2D Images

The earliest example of encoding objects with CPPNs occurred in 2D space. In order to evolve images as in the original Picbreeder [10], CPPNs generate a color for each pixel in an image. Specifically, CPPNs take the current x and y pixel coordinates, the distance from the image center, and a constant bias as input. The distance input is not necessary to uniquely specify a pixel, but allows CPPNs to easily create radial patterns. Image coordinates are scaled to the range $[-1, 1]$ along the x and y axes before input values are calculated.

The CPPN outputs are hue, saturation, and brightness (HSB) for the pixel at the given coordinates. HSB color space is used because the image shape only depends on brightness, whereas the color depends on hue and saturation. Separation of these components allows more stable progression across generations, because mutations affecting only the shape or only the color are common.

HSB values must be in the range $[0, 1]$, so the saturation output is clamped to this range. However, final pixel brightness is the absolute value of the brightness output clamped to this range, so there is only a narrow range where crisp dark lines are rendered. The hue output is clamped to $[-1, 1]$, but then 1 is added to negative values in order to derive the actual pixel hue. Allowing negative values in the first step assures that activations transitioning between positive and negative vary smoothly across a rainbow band of color.

The version of Picbreeder used in this paper is different from the original version in that it is not a multi-user online collaborative platform. However, this system can generate the exact same images as the original program. Genomes from the original Picbreeder were

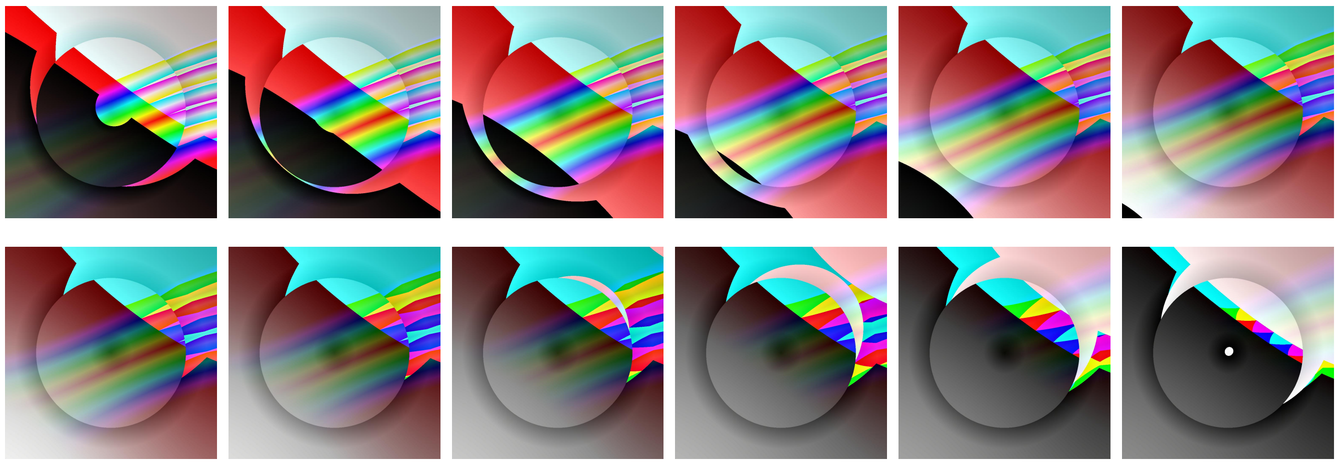


Figure 1: Selected Frames Generated by a CPPN Evolved Using AnimationBreeder. Images from a particular 2D animation are shown at two-frame intervals. The images highlight how the time input to a CPPN transforms an image over time. The complete, smooth animation can be seen online at https://people.southwestern.edu/~schrum2/SCOPE/EvolvedArt/cppnart.php along with several other animations generated in the human subject study described in Section 5.

made available as part of the CPPN Explorer project [4], and when converted to be compatible with the version of Picbreeder in this paper, identical images were produced. In fact, this system also supports many activation functions not available in the original Picbreeder, which are listed in Section 4. These additional functions increase the expressive power of the evolved CPPNs.

3.4 Evolving 2D Animations

A technical contribution of this work is the extension of image generating CPPNs into animation generating CPPNs. In the past, CPPNs have been queried exclusively in spatial coordinate spaces to create static objects, but CPPNs can also be queried along a time coordinate. Animations are created by having CPPNs generate different images for each frame of animation. The program that evolves CPPNs with this approach is called AnimationBreeder. Selected frames of an example animation are shown in Fig. 1.

These CPPNs have the same outputs and inputs as Picbreeder, as well as a new time input so that distinct images can be generated at each frame of animation. As the time input increases, the output of the CPPN changes, producing an animation. In order to generate smooth transitions, the difference between subsequent time inputs must be sufficiently small. Therefore, a standard of 24 frames per second was used to determine the difference between subsequent time inputs: $1/24$. This frame rate is standard for most animation and ensures smooth motion in generated animations. The first time input for an animation is 0, but there is no bound on the growth of the time input. Therefore, CPPNs can generate animations of arbitrary length by increasing the time input enough times in small increments of $1/24$, but lengthening the animation also increases the amount of time it takes to generate the animation.

3.5 Evolving 3D Shapes

Beyond evolving 2D artifacts, this paper also studies programs that evolve 3D artifacts. The program that evolves static 3D shapes is based off of Endless Forms [2], but is called 3DObjectBreeder for clarity, and because of slight differences between the two programs.

To evolve shapes with a CPPN, inputs for x, y, and z coordinates designating specific voxels within a $10 \times 10 \times 15$ volume are used. The distance of each voxel from the center of the cuboid is also provided, along with a constant bias. The voxel is determined to be present if a designated output exceeds a threshold of 0.1, in which case a cube is rendered at the designated location. The combination of cubes results in interesting shapes.

The process above allows for the evolution of 3D shapes in a manner similar to the original Endless Forms. However, Endless Forms also applied the Marching Cubes algorithm [7] to smooth the blocky shapes encoded by the CPPNs. This smoothing enhancement is not yet applied in 3DObjectBreeder, though such functionality could be added in the future. Currently, the generated shapes are blocky, but still visually interesting. Some enhancements also distinguish 3DObjectBreeder from the original Endless Forms.

Specifically, 3DObjectBreeder expands on the capabilities of the original Endless Forms in the two ways. First, each voxel in 3DObjectBreeder can have its own color by obtaining HSB values from designated CPPN outputs. The addition of color variation between voxels has been explored previously [6], but not using interactive evolution. 3DObjectBreeder's second enhancement over Endless Forms is the ability of each voxel to be displaced slightly from its prescribed location. As a result, voxels are not confined to a rigid grid. Specifically, the CPPN has separate outputs for displacing a voxel along the x, y, and z axes. The maximal displacement allowed pushes the centroid of the voxel to the original voxel boundary. In other words, CPPN outputs for displacement are scaled to be within $[-v/2, v/2]$ where $v = 10$ is the length of each edge of a voxel.

The above procedure generates static 3D shapes in a manner distinct from Endless Forms, yet interesting in its own right. As in Endless Forms, the 3D shapes rotate so that they can be viewed from multiple angles, but despite this animation, the shapes themselves are static. However, the actual representation of the 3D shapes can also be animated.

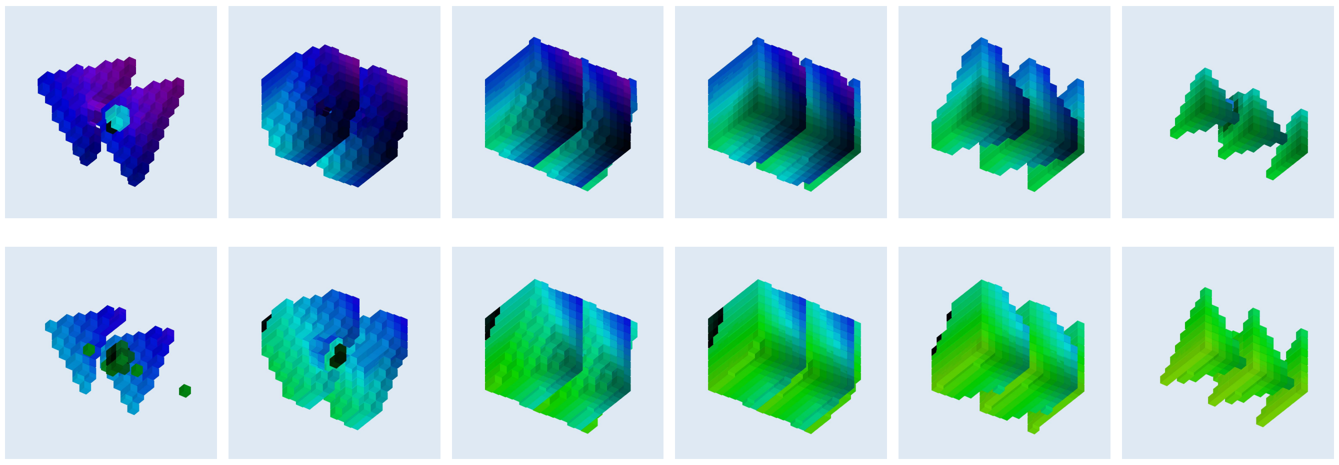


Figure 2: Selected Frames Generated by a CPPN Evolved Using 3DAnimationBreeder. Images from a particular 3D animation are shown at two-frame intervals. The images highlight how the time input to the CPPN adds and subtracts voxels from a shape over time, in addition to changing their colors. Voxels can also wiggle within certain bounds. The complete, smooth animation can be seen at https://people.southwestern.edu/~schrum2/SCOPE/EvolvedArt/cppnart.php along with several other animations generated in the human subject study described in Section 5.

3.6 Evolving 3D Animations

To create 3D animations, the same method was applied that was used to extend Picbreeder into AnimationBreeder. A time input was added to each CPPN, so that CPPNs could be queried across time to create multiple outputs displayed at 24 frames per second. The program that generates these animations is called 3DAnimationBreeder. Selected frames of an example animation can be seen in Fig. 2.

Because the CPPN outputs determine the presence/absence of voxels, and also their displacement, the resulting animations often contain interesting visual rippling effects or cyclic construction and deconstruction of the shape. The colors of voxels can also change, further enhancing the allure of the animations. Although all of these aspects of the shape can vary at once, it can be particularly interesting to see certain aspects changing while others remain fixed or vary only subtly. Static shapes can have interesting color patterns dancing along their surfaces, and shapes with solid colors can undulate or disintegrate before the user's eyes. If the user prefers, then all of these aspects can vary wildly at once, creating chaotic results.

Because the animations increase the amount of activity in the visualizations, the 3D animations do not rotate as the static 3D shapes do, though rotations could easily be enabled. Therefore, all perceived movement is purely due to the CPPN. However, the user interface does allow 3D animations to be viewed from different angles, as described in the next section on the user interface.

4 USER INTERFACE

The interface is based on those used in Picbreeder and Endless Forms. Every interface includes an Evolve button, which is how users advance to the next generation. Users select and unselect items, then click Evolve to go to the next generation. There is also an Undo button to go back one generation, a Restart button for initializing a new population, and a Save button to save a copy of an evolved artifact. All interfaces also contain a slider for changing the number of mutation chances per offspring on a scale of 1 to 10. This slider offers better control over evolution: the low end allows users to finely control evolution by making only small changes to the population, while the high end allows the users to explore the search space more rapidly and evolve distinct artifacts in fewer generations. These features are sufficient to interactively evolve novel artifacts, but the interface also includes additional features that give users more control over artifact evolution.

One addition is checkboxes corresponding to each available activation function. By clicking or unclicking a box, the specified activation function is added to or removed from the set of activation functions that can be introduced when mutations that splice a new node or change an existing activation function occur. The activation functions available on the interface are sigmoid (full and half), Gaussian (full and half), cosine, sine, identity (unbounded, and clamped to $[0, 1]$ and $[-1, 1]$), absolute value, square wave, triangle wave, sawtooth (full and half), and rectified linear units. Full versions of functions are stretched to the range $[-1, 1]$, while half versions are confined to $[0, 1]$. These additional functions make the emergence of patterns not seen in previous work possible, which is especially true of the functions with sharp corners and discontinuities.

All interfaces also include a button to view the CPPNs generating each artifact instead of the artifacts themselves. Viewing the CPPNs allows users to see differences between not just the artistic phenotypes, but also their genotypes. A user can therefore focus on the presence of particular activation functions and/or structures.

Finally, each interface has checkboxes corresponding to each input in the CPPN. Clicking and unclicking these boxes enables and disables the chosen inputs (Fig. 3). Disabled inputs have a fixed value of 0 every time the CPPN is queried, allowing the user to explore the effect of each input on the phenotype. Despite the depth of analysis in both the original Picbreeder paper [10] and the recent CPPN Explorer paper [4], this input-disabling option is a new way to understand how CPPNs construct interesting artifacts.

The above features are common to all programs developed in this paper. On top of these features, the AnimationBreeder, 3DAnimationBreeder and 3DObjectBreeder interfaces each have additional

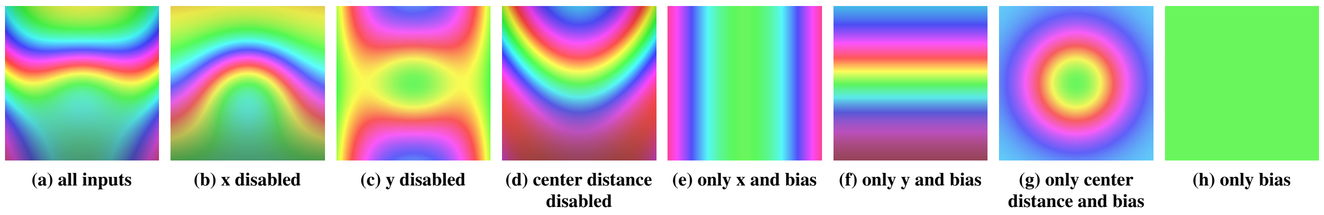


Figure 3: Same CPPN Generating Images With Different Inputs. In all interactive evolution interfaces, the user can disable CPPN inputs, so that they use a default value of 0 whenever queried. All of these images are generated by the same CPPN with (a) all CPPN inputs, (b) x-coordinates disabled, (c) y-coordinates disabled, (d) distance from center of image disabled, (e) only x-coordinates and the bias enabled, (f) only y-coordinates and the bias enabled, (g) only distance from center of image and the bias enabled, and (h) only the bias enabled. Exploring the input space to the CPPNs not only helps to generate new interesting results, but also allows one to better understand how the geometry of the input space affects the CPPN outputs. The input space of 3D and animated artifacts can be explored in a similar manner.

features. In the two animation programs, there is a slider that controls the length of the animation from 10 to 250 frames. Another slider allows the user to add pauses between frames to facilitate viewing of each individual frame. This slider ranges from 10 to 50 milliseconds. Finally, another slider controls the pause between the end of an animation and the next playback, since animations cycle repeatedly. This slider also ranges from 10 to 50 milliseconds. These options allow users to better inspect their animations.

The 3DObjectBreeder interface contains several special features. Although 3DObjectBreeder evolves static shapes, the interface depicts an animation of the shapes rotating. Therefore, a slider controlling the pause between frames is included. There is also a dropdown to assign a fixed color to all evolved shapes as in Endless Forms, or use the evolved CPPN to define voxel coloring as described in Section 3.5. Another dropdown selects between vertical and horizontal rotation axes, to allow better views of the evolved shapes.

The 3DAnimationBreeder also has special features. Because its animations do not rotate, there are sliders that adjust the fixed angle of vertical and horizontal rotation the animation is viewed from.

Overall, the interface helps users explore their creations and pick the most pleasing artwork to evolve. Videos of interaction with the interfaces are at http://people.southwestern.edu/~schrum2/SCOPE/EvolvedArt/cppnart.php, and source code can be downloaded from https://github.com/schrum2/CPPNArtEvolution.

The rich interfaces described thus far allow an experienced user to have a great deal of control over the evolution of interesting artifacts. However, in order to directly study the benefit of introducing a time input to generate animations, a simplified interface is used for the purposes of a human subject study.

5 HUMAN SUBJECT STUDY

This section describes the procedure and results of the human subject study comparing still and animated programs.

5.1 Procedure

To gauge user response to these newly created systems, a human subject study was conducted in which 40 people interacted with the systems and gave feedback on a survey afterward. There were two different studies, each with 20 distinct participants. In the first study, users interacted with Picbreeder and AnimationBreeder. In the second study, users interacted with 3DObjectBreeder and 3DAnimationBreeder. Advanced features of the interface were disabled for the study. Only the slider controlling mutation chances and the Evolve button for progressing to the next generation were available.

The following settings were common across all four programs. Users interacted with each program for fifteen generations, with a population of 20 CPPNs. The activation functions available to the CPPNs were the sigmoid (half), Gaussian (half), sine, sawtooth (half and full), identity clamped to $[0, 1]$, triangle wave, and square wave functions. For each new offspring, the chance of crossover was 50%. Once created, the offspring could have 1 to 10 chances of mutation, depending on the current slider setting chosen by the user. For each mutation chance, the rate of activation function change was 30%, the per link weight perturbation rate was 5%, the link creation rate was 40%, and the node splice rate was 20%.

For AnimationBreeder and Picbreeder, feature selection was enabled, meaning that each initial CPPN had only one incoming link per output neuron [14]. Feature selection makes the initial pictures and animations very simple, which means that users can generally observe a large increase in complexity over the course of a session. For AnimationBreeder, the animation length was 50 frames.

For 3DObjectBreeder and 3DAnimationBreeder, the animation length was 72 frames and feature selection was disabled, so that every input neuron connected to every output neuron. Feature selection was disabled in the 3D programs because it hindered evolution too much, by making it difficult to evolve beyond boring initial shapes. The number of animation frames is different because 3DObjectBreeder requires 72 frames of animation to produce a smooth, 360 degree rotation of evolved shapes. Although 3DAnimationBreeder does not display rotating shapes, it also uses 72 frames of animation for fair comparison with the 3DObjectBreeder.

In each session, subjects evolved artifacts for 15 generations with one program, and then 15 generations with the other program. Half of the users were exposed to an animating program first and half used the animating program second, to ensure that the order of exposure did not systematically affect the results. After interaction with both programs was completed, users filled out a survey that compared the two programs. They were asked which program they preferred, and to compare and describe their favorite final results from each program. They rated their enjoyment of each program, the complexity of their final results, and how aesthetically pleasing their final results were on a scale of 1 to 5. They were also asked to briefly describe their thought process when selecting artifacts to evolve to subsequent generations.

5.2 Quantitative Results

In both the 2D and 3D studies, more users preferred animated programs and results over the still counterparts (Table 1). Over three times as many users preferred AnimationBreeder over Picbreeder, but only one more user preferred the final AnimationBreeder result over the final Picbreeder result. For 3D programs and results, at least twice as many users preferred animation over the still counterparts.

Samples for the individual 20 user studies are too small to compute statistically significant results. Additionally, users could indicate equal preference for both options, but it is unclear what value to assign to this option under a null hypothesis. Therefore, results from both studies are pooled into a single sample of size 40 to determine preference between still and animated programs, and a sample of size 39 to determine preference between still and animated final results (one user failed to answer this question on the survey). Ties are split evenly between the two choices, with extra votes given to the still programs, since this approach is more conservative than discarding tie votes. These processed results are compared with a one-sided binomial test, which indicates that users significantly prefer animated programs over still programs ($p \approx 0.01924$), but that there is no statistically significant difference in preference for final animated results over final still results ($p \approx 0.2612$).

Table 2 shows users' rankings of their enjoyment of each of the four programs. Users enjoyed animated programs more than still programs, but generally gave high marks to both. When the 2D and 3D results are pooled to create a larger sample ($n_1 = n_2 = 40$), a Wilcoxon-Mann-Whitney U Test reveals that the animation programs are ranked higher than the still programs by a statistically significant amount ($U = 602, p \approx 0.04141$). To account for the high number of ties in the data, the EDISON-WMW algorithm [8] was used to compute precise p-values.

Table 3 displays users' rankings of the complexity of their final results from each of the four programs. The results show a wide variety of ratings across each of the programs, though most tend toward middling ratings. However, pooling the 2D and 3D results reveals that users rank the 3D results as more complex by a statistically significant amount ($U = 547.5, p \approx 0.011205$).

Table 4 shows user ratings for the aesthetic appeal of their final results. Interestingly, most users found still 2D images more appealing than 2D animations. However, 3D animations were more appealing than still 3D shapes. These opposite outcomes for the 2D and 3D comparisons mean that there is no significant difference between still and animated aesthetic appeal in the pooled comparison ($U = 815, p \approx 0.87871$).

In summary, the quantitative results indicate that users prefer animation programs and find them more enjoyable. They also find the results of the animation programs more complex, but not necessarily more aesthetically appealing. Their preference between the final animations and final static results is unclear. Analysis of the qualitative results gives further insight into these quantitative findings.

Table 1: User Preferences.

Comparison	Still	Animated	Equal
2D programs	15%(3)	55%(11)	30%(6)
2D results	45%(9)	50%(10)	5%(1)
3D programs	25%(5)	60%(12)	15%(3)
3D results	26.3%(5)	52.6%(10)	21.1%(4)

User preferences are shown for still and animated programs, and the final favorite results produced with these programs. Users could indicate that they favored one option over the other, or that they had equal preference for both. The integers in parentheses indicate the number of people who chose each response. For the comparison of 3D results, one user failed to answer the question, so percentages are computed out of 19 responses, rather than 20.

Table 2: User Enjoyment

Rating	1	2	3	4	5
2D still	0%(0)	5%(1)	5%(1)	60%(12)	30%(6)
2D animated	0%(0)	0%(0)	10%(2)	30%(6)	60%(12)
3D still	10%(2)	0%(0)	30%(6)	35%(7)	25%(5)
3D animated	0%(0)	5%(1)	15%(3)	40%(8)	40%(8)

User ratings of their enjoyment of each program on a scale of 1 to 5, showing both percentages and exact numbers. Both types of animations are preferred over still artifacts, though 2D programs are rated higher than 3D programs.

Table 3: Complexity of Final Results

Rating	1	2	3	4	5
2D still	5%(1)	10%(2)	45%(9)	25%(5)	15%(3)
2D animated	5%(1)	5%(1)	10%(2)	45%(9)	35%(7)
3D still	0%(0)	30%(6)	50%(10)	5%(1)	15%(3)
3D animated	5%(1)	20%(4)	15%(3)	40%(8)	20%(4)

User ratings of the complexity of their favorite final result from each program on a scale of 1 to 5, showing both percentages and exact numbers. Animations are found to be more complex that their still counterparts.

Table 4: Aesthetic Appeal of Final Results

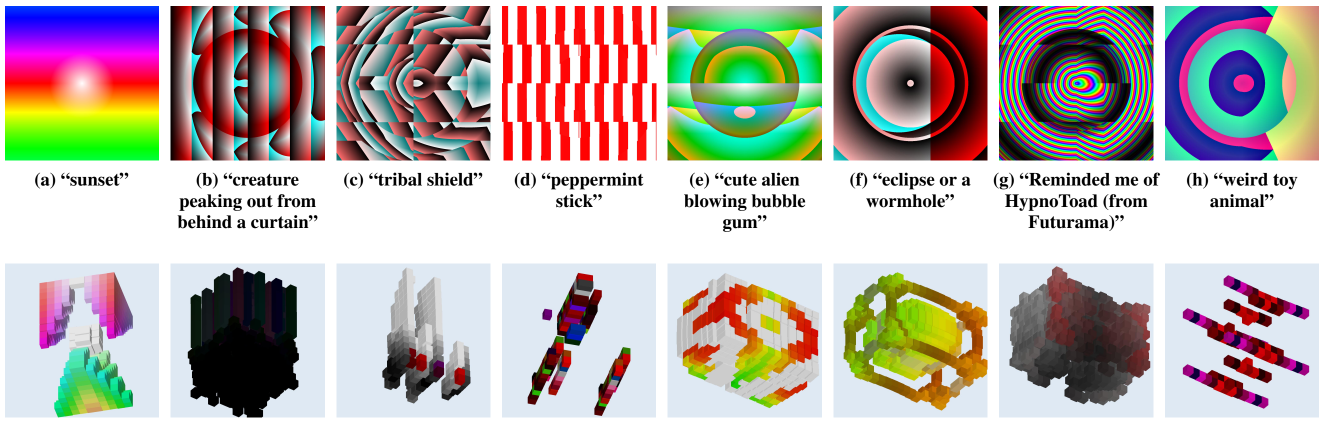
Rating	1	2	3	4	5
2D still	0%(0)	5%(1)	5%(1)	25%(5)	65%(13)
2D animated	0%(0)	5%(1)	10%(2)	50%(10)	35%(7)
3D still	0%(0)	5%(1)	20%(4)	45%(9)	30%(6)
3D animated	5%(1)	0%(0)	5%(1)	40%(8)	50%(10)

User ratings of the aesthetic appeal of their favorite final result from each program on a scale of 1 to 5, showing both percentages and exact numbers. Although 3D animations are more appealing than still 3D shapes, users found still 2D images to be more aesthetically appealing than 2D animations.

5.3 Qualitative Results

Users were asked to describe what their favorite result from the final generation of each program looked like. Many users described the specific color and movement patterns, but some saw interesting shapes and images within the artifacts. For the three-dimensional programs, users said they were reminded of items such as "a fleet of spaceships" (Fig. 4l), a "castle on a hill" (Fig. 4j), "cake", and "a beating heart" (Fig. 4o). However, most of the 3D objects and animations were described as colorful cuboids or a construction of tiny cubes that moved around. Because Marching Cubes was not used to smooth the final shapes, the 3D objects look blocky because the individual cubes used to construct the objects are visible.

For the two-dimensional programs, interesting shapes were more readily noticed. Some notable descriptions are a "sunset" (Fig. 4a), a "creature peeking out from behind a curtain" (Fig. 4b), a "tribal shield" (Fig. 4c), a "peppermint stick" (Fig. 4d), a "cute alien blowing bubble gum" (Fig. 4e), an "eclipse or wormhole" (Fig. 4n), "HypnoToad from Futurama" (Fig. 4g), and "a weird toy animal that pops out



(a) "sunset" **(b) "creature peaking out from behind a curtain"** **(c) "tribal shield"** **(d) "peppermint stick"** **(e) "cute alien blowing bubble gum"** **(f) "eclipse or a wormhole"** **(g) "Reminded me of HypnoToad (from Futurama)"** **(h) "weird toy animal"**

(i) shape cracked in two **(j) "castle on a hill"** **(k) "four right triangles"** **(l) "fleet of space ships"** **(m) "moving gradient of colors"** **(n) shrinking orb in a cage** **(o) "beating heart"** **(p) growing and shrinking red sticks**

Figure 4: Selected Items Generated by Human Subjects. All items shown were generated by users in the human subject study. Images (a)–(d) show still images from Picbreeder. Images (e)–(h) are individual frames from different animations evolved by AnimationBreeder. Images (i)–(l) are shapes evolved by 3DObjectBreeder shown from a particular point of view. Images (m)–(p) are individual frames from different animations evolved by 3DAnimationBreeder. Double quotes indicate the text was provided by a human subject. These images show the diversity of results that different users produced, and the user descriptions demonstrate how even abstract art can evoke imaginative descriptions with references to real-world objects.

at you" (Fig. 4h). Many described their images and animations as resembling eyes and rainbows. Circular shapes are easy to generate thanks to the distance from center input given to CPPNs. Rainbow patterns are a side effect of using the HSB color encoding, because as hue varies, the colors of the rainbow are traversed.

Most of the users said that they enjoyed the programs overall, and liked being able to breed certain patterns over time based on the images that they clicked, but some people were disheartened by the long loading time for the animations. One user in particular said that the study was "fun but took forever". Having a slow-loading system can contribute to user fatigue. This long loading time is not indicative of interacting with the program regularly. The human subjects experienced a slower version of the programs because of the need to save all of the artifacts that users clicked for each generation, which slowed down the programs considerably. Some users also said that the three-dimensional animations made them feel physically uncomfortable and strained their eyes. This is a potential cause for concern and could contribute to user fatigue as well, but was to be expected because of the disjointed movements and flashing colors of some of the animations, and because of the fact that so many animations are displayed on the interface simultaneously.

6 DISCUSSION AND FUTURE WORK

The art interactively evolved by human subjects and their responses to survey questions show that introducing time to evolve animated versions of artistic artifacts results in complex and enjoyable results. Creating animated artifacts from still artifacts adds a new element of intrigue and complexity that builds upon the original Picbreeder and Endless Forms, thus opening new avenues of exploration for interactive evolution using CPPNs.

Although users preferred using the animation generating programs, they sometimes preferred their static final results more then their animations. User comments provide potential explanations for these findings. For example, some users lost art that they enjoyed more than the results in their final generation because they stopped selecting their preferred art in favor of new, novel items. If the users were not keeping track of the number of generations passed, the program could terminate suddenly with potentially displeasing results for the user. Therefore, users may have enjoyed the many animated results that were produced more than the static results, but were coincidentally left with final animated results that they did not appreciate as much. An extreme case of this is a user that was intrigued by the unexpected appearance of an completely empty space when evolving 3D animations. Unfortunately, the user selected only this option, and was thus left with a final result that, while novel, was neither complex nor pleasing. A better understanding of how the system works, specifically the fact that selection is elitist, could remedy this problem. The Undo button available in the full program interface would also help address this problem.

In the 2D case specifically, users found the still results more aesthetically appealing. This outcome may be related to the use of feature selection [14] in network initialization. If the time input is underutilized, then AnimationBreeder will produce still images, or animations with very little movement. There are several users whose final AnimationBreeder results are not animated. Starting evolution with fully connected networks, as was done with the 3D programs, would increase the occurrence of interesting animations.

Although some users likened their creations to real-world objects, observation of all final results indicates that most of the art produced is quite abstract, in contrast to prominent results from the original Picbreeder and Endless Forms. This discrepancy arises because Picbreeder and Endless Forms are online and collaborative, so multiple users can further evolve the results of others. In contrast, there is a limit to what a single user can evolve within 15 generations. However, the use of genomes from the CPPN Explorer project [4] validates the expressive capacity of the CPPNs used in this paper, and extended use of these programs by the authors has led to many interesting artifacts more complex than what users from the study produced. However, these users still found interesting ways to describe the artifacts they generated (detailed in Section 5.3).

Some users were overwhelmed when evolving animated artifacts. Because there were 20 buttons containing moving multicolored artifacts, AnimationBreeder and 3DAnimationBreeder are highly visually stimulating. In fact, some users thought that the animated programs were too visually stimulating and strained their eyes, so an option was added to the code after the human subject study restricting animations to only occur on the item the user's mouse is hovering over. Some participants got worn out by occasionally long waiting times between generations. Although user fatigue is a common issue in interactive evolution, this issue was exacerbated in the study by the need to save all results produced by the users, and is not an inherent problem with the programs. Although there are ways that the code could be optimized to be faster, waiting times are generally not a problem on modern machines.

As previously discussed, the 3D artifacts of this paper were different from those of the original Endless Forms because they remain blocky instead of being smoothed. If the code were adapted to make use of hardware accelerated graphics libraries, then the application of the Marching Cubes algorithm for smoothing would be relatively straightforward. It would be interesting to see how the Marching Cubes algorithm would affect 3D animations. Animations might also be enhanced by the use of shaders for both 3D and 2D animations.

Incorporation of sound into generated animations might also produce interesting results. Specifically, a sound wave could be used with or instead of a time input to generate a 2D or 3D animation, so that the animation is synchronized with the sound, resulting in interesting audio visualizations. An audio signal can be thought of as a function of time, but other functions of time could also be used as CPPN inputs. Alternatively, one CPPN could generate a sound (as with Breedesizer [5]) in addition to an animation, which would provide another way to synchronize audio and visual components. In short, there are many ways to use the concept of time to augment artistic generation by CPPNs.

There are also possible applications beyond artistic artifacts. Time can be incorporated into the CPPN generation of soft robots [1] to perhaps introduce a form of aging. Neural networks generated by CPPNs via HyperNEAT [12] could also depend on a time input, so that the substrate network defining the policy of an agent in a sequential decision making problem could change throughout the course of evaluation. Having a soft body or policy that changes during an agent's lifetime could be useful in domains where goals or the environment change in drastic ways throughout the course of evaluation. In fact, there could even be CPPN inputs that are derived from the phenotype network making decisions in the environment, which would allow network structure to adapt not only over time, but in response to specific environmental stimuli. This notion has connections to the concept of neural plasticity, which have already been explored using HyperNEAT [9]. There is also a related approach, known as HyperNetworks [3], that have been used to generate recurrent networks whose weights change over time in order to generate text and handwriting sequences. These general ideas could also be applied in sequential decision making problems.

7 CONCLUSION

CPPNs are capable of generating diverse original images and objects. However, time had not previously been incorporated into interactive evolution to create animating images and moving objects. Adding time as an input to a CPPN allowed previously established programs (Picbreeder and Endless Forms) to be expanded on in new and exciting creative ways, generating results that are enjoyable to watch and visually complex. According to a human subject study that directly compared these animated programs to their still counterparts, users found the animated programs to be more enjoyable than the still programs. Despite some issues with the graphical interface, the new animated programs, AnimationBreeder and 3DAnimationBreeder, produce results that are visually interesting and intricate. In the future, time can be applied to other applications of CPPNs, such as soft robots and neural networks.

ACKNOWLEDGMENTS

The authors would like to thank Joost Huizinga for his help understanding the Picbreeder genomes from the CPPN Explorer project. This research is supported in part by the Summer Collaborative Opportunities and Experiences (SCOPE) program for undergraduates, funded by various donors to Southwestern University.

REFERENCES

[1] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2013. Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. In *Genetic and Evolutionary Computation Conference*.

[2] Jeff Clune and Hod Lipson. 2011. Evolving Three-dimensional Objects with a Generative Encoding Inspired by Developmental Biology. In *European Conference on Artificial Life*. 141–148.

[3] David Ha, Andrew Dai, and Quoc V. Le. 2017. HyperNetworks. In *International Conference on Learning Representations*.

[4] Joost Huizinga, Kenneth O Stanley, and Jeff Clune. 2017. The Emergence of Canalization and Evolvability in an Open-Ended, Interactive Evolutionary System. *arXiv preprint arXiv:1704.05143* (2017).

[5] Björn Þór Jónsson, Amy K. Hoover, and Sebastian Risi. 2015. Interactively Evolving Compositional Sound Synthesis Networks. In *Genetic and Evolutionary Computation Conference*. 321–328.

[6] Joel Lehman, Sebastian Risi, and Jeff Clune. 2016. Creative Generation of 3D Objects with Deep Learning and Innovation Engines. In *International Conference on Computational Creativity*.

[7] William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Conference on Computer Graphics and Interactive Techniques*. 163–169.

[8] Alexander Marx, Christina Backes, Eckart Meese, Hans-Peter Lenhof, and Andreas Keller. 2016. EDISON-WMW: Exact Dynamic Programing Solution of the Wilcoxon-Mann-Whitney Test. *Genomics, Proteomics & Bioinformatics* 14, 1 (2016), 55–61.

[9] Sebastian Risi and Kenneth O. Stanley. 2012. A Unified Approach to Evolving Plasticity and Neural Geometry. In *International Joint Conference on Neural Networks*. 1–8.

[10] Jimmy Secretan, Nicholas Beato, David B. D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T. Folsom-Kovarik, and Kenneth O. Stanley. 2011. Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation* 19, 3 (2011), 373–403.

[11] Kenneth O. Stanley. 2007. Compositional Pattern Producing Networks: A Novel Abstraction of Development. *Genetic Programming and Evolvable Machines* 8, 2 (2007), 131–162.

[12] Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. 2009. A Hypercube-based Encoding for Evolving Large-scale Neural Networks. *Artificial Life* (2009).

[13] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation* 10 (2002), 99–127.

[14] Shimon Whiteson, Peter Stone, Kenneth O. Stanley, Risto Miikkulainen, and Nate Kohl. 2005. Automatic Feature Selection in Neuroevolution. In *Genetic and Evolutionary Computation Conference*. ACM.