DrawCompileEvolve: Sparking Interactive Evolutionary Art with Human Creations

Jinhong Zhang[1], Rasmus Taarnby[1], Antonios Liapis[2], and Sebastian Risi[1]

[1] Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
[2] Institute of Digital Games, University of Malta, Msida, Malta
jinh@itu.dk, reta@itu.dk, antonios.liapis@um.edu.mt, sebr@itu.dk

Abstract. This paper presents *DrawCompileEvolve*, a web-based drawing tool which allows users to draw simple primitive shapes, group them together or define patterns in their groupings (e.g. symmetry, repetition). The user's vector drawing is then compiled into an *indirectly* encoded genetic representation, which can be evolved interactively, allowing the user to change the image's colors, patterns and ultimately transform it. The human artist has direct control while drawing the initial seed of an evolutionary run and indirect control while interactively evolving it, thus making DrawCompileEvolve a mixed-initiative art tool. Early results in this paper show the potential of DrawCompileEvolve to jump-start evolutionary art with meaningful drawings as well as the power of the underlying genetic representation to transform the user's initial drawing into a different, yet potentially meaningful, artistic rendering.

1 Introduction

In evolutionary art and music, interactive evolutionary computation is often the method of choice for exploring the search spaces of such highly subjective domains [1–7]. Human users are uniquely capable of detecting visual or aural patterns and associating them with real-world stimuli. Moreover, the individual tastes of each human user can drive evolution to focus on different areas of the search space that are uniquely valuable to the current user. The particular imagination, preference and real-world experience of a human user can guide search in interesting, unexpected and meaningful ways.

Often, the genetic representation in evolutionary art and music allows for a vast expressive range and therefore an expansive search space. In order for human users to navigate such space, extensive interaction with the system may be necessary, leading to user fatigue [8]. Moreover, when a human explores the search space without a purpose or a tether, their lack of aim or context makes the evaluation of artifacts less meaningful.

This paper introduces *DrawCompileEvolve*, a web-based drawing tool which allows human users to seed the initial population of their evolutionary search with their own drawings. DrawCompileEvolve allows users to draw certain primitive shapes on a canvas, and define regularities between them such as symmetry and repetition; these primitives and patterns are compiled into an indirect representation called *computational pattern-producing network* (CPPN; [9]) and can be subsequently evolved interactively building on the Picbreeder art application [1]. Importantly, because the compiled

CPPNs directly embody the annotated regularities, the produced offspring shows meaningful variations (e.g. symmetrically defined body parts exhibit a coordinated change in size). The expressive range of a CPPN, its ability to effectively encode repetition, symmetry and elaboration [9], and the locality in its search (compared to e.g. genetic programming) makes it uniquely suited for the presented mixed-initiative art tool.

By starting the interactive evolution of images from a semantically meaningful or aesthetically preferable area of the search space, DrawCompileEvolve allows for more control over the evolutionary process. Moreover, as early results in this study show, users of DrawCompileEvolve are more likely to use interactive evolution to transform their initial drawings to other types of meaningful imagistic representations.

2 Background

DrawCompileEvolve compiles vector graphics into CPPN-encoded images via a method described in Section 5; resulting CPPN images are interactively evolved via CPPN-NEAT, using the Picbreeder evolutionary art tool. Details on CPPNs and Picbreeder are provided in Section 2.1, while interactive evolution is discussed in Section 2.2.

2.1 CPPN-NEAT

A compositional pattern producing network (CPPN) is, in essence, an artificial neural network (ANN) with a wider variety of activation functions [9]. The activation functions of a CPPN include Gaussian and sine functions, which produce patterns such as symmetry and repetition. This allows a CPPN to recreate and describe underlying structures found in nature as a result of an evolutionary process but without the need of simulating this process [9, 1]. The patterns in the CPPN's outputs are affected by the inputs of the network, by the activation functions of its nodes and by the weights of connections between these nodes. The pattern-producing capability of CPPNs has been exploited to create images [1], three-dimensional models [2], artificial flowers [3], spaceships [4], particle effects [5], musical accompaniments [6] and drum tracks [7]. CPPNs can be evolved via *neuroevolution of augmenting topologies* (NEAT), which traditionally starts from minimal topology networks and adds nodes and links [10]. As NEAT adds new nodes on top of existing pattern-producing nodes, it is very likely that the underlying patterns of smaller networks (e.g. symmetry) will be retained in the larger networks of their offspring. In this paper, instead of starting from a minimal topology, the initial CPPNs (which encode images) are created based on the user's vector drawing.

Among the different implementations of CPPN-NEAT, DrawCompileEvolve makes heavy use of Picbreeder [1]. Picbreeder is an online service where users can collaboratively evolve images and describes itself as a "Collaborative Interactive Art Evolution" program. The two-dimensional images in Picbreeder and in this paper are created by providing the coordinates of each pixel in the image as input to the CPPN and extracting the color of that pixel from the CPPN's output. The program displays the images of the current generation to the user, who can guide evolution by selecting one or more images; those images breed to create the next generation. Users can start "from scratch" with a minimal topology network, or can start from another user's saved creation.

2.2 Interactive Evolution

Interactive evolutionary computation (IEC) uses human feedback to evaluate content; essentially, with IEC human users decide which individuals breed and which ones die. As surveyed by Takagi [8], IEC has been applied to several domains including games, industrial design and data mining. Evolutionary art and music often hinges on human evaluation of content due to the subjective nature of aesthetics and beauty. However, constant requirement for feedback and the cognitive overload of evaluating a broad range of content can result in *user fatigue*. Several steps have been taken to lessen user fatigue by IEC methods, such as limiting the rating levels of fitness, predicting fitness from user rankings [11], or showing a subset of the evolving population to the user.

Another potential solution to the problem of user fatigue is seeding the initial population of IEC with meaningful, high-quality individuals. By ensuring that the starting point of evolution is tailored to the problem at hand, the user needs to spend less time exploring the space of possible solutions; the initial seed largely defines the area of the search space that should be explored. More importantly, however, using a personalized initial seed which is appealing or meaningful to the current human user of IEC enriches the interaction. As Takagi puts it: "IEC users evaluate individuals according to the distance between the target in their psychological spaces and the actual system outputs, and the EC searches the global optimum in a feature parameter space according to the psychological distance" [8]: an initial seed that resonates either semantically (e.g. a depiction of a loved one) or aesthetically (e.g. a desired color palette) with the user ensures that the system output will have a high value in the user's psychological space.

Several attempts at seeding interactive evolution have been made: superficially, choosing the right genotypical representation is a form of seeding as well. Seeding IEC can be achieved by starting from previous users' creations, e.g. in saved, previously evolved images of Picbreeder [1]. Endless Forms allows users to upload any three-dimensional model, the shape of which is then transformed by CPPNs but yet recognizable due to its input matching the uploaded model closely [12]. Similarly, Maestro-Genesis indirectly seeds evolution by using a human-authored sound file as a scaffold to generate musical accompaniments through IEC [6]. Finally, Sentient World seeds evolution by using back-propagation to generate game terrain that matches a low-level sketch drawn by the user [13].

3 DrawCompileEvolve System Architecture

DrawCompileEvolve is a web-based tool which allows users to create shapes via a traditional draw interface, then evolve their vector drawings as rasterized images encoded by CPPNs. The tool is available at http://rasmustaarnby.dk/drawcompileevolve. The individual components of the tool will be described in the next sections.

The overall structure and the relation between the different parts of the program are depicted in Figure 1. The system consists of three layers: frontend, content management system (CMS) and backend. The frontend handles the drawing interface and displays the user's profile and the other users' creations. The CMS system *WordPress* acts as a separation layer between the frontend and the backend, leaving the Picbreeder backend intact while retaining full access to the database. The backend is divided into the

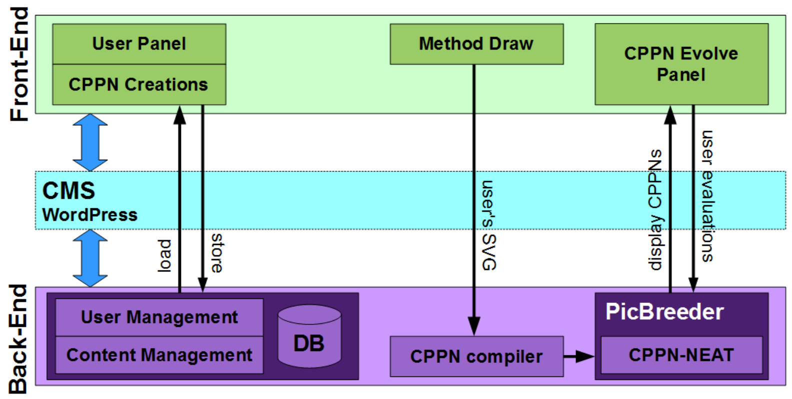


Fig. 1: System architecture of DrawCompileEvolve.

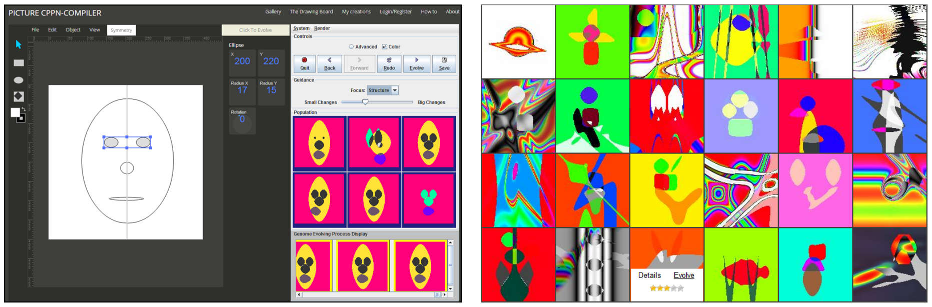
database which holds all website content, user information and data about the different CPPN creations, and the Picbreeder applet which is responsible for compiling the user created shapes (in scalable vector graphics format) into a CPPN, evolving the CPPN and rendering the images into viewable JPG format.

4 Drawing Interface

A core motivation of DrawCompileEvolve is allowing users, regardless of technical proficiency, to create CPPN-encoded images. This is achieved via an online web application, which lets users log in and draw images directly in the browser; the images are then automatically transferred into an evolvable genotypic representation by the CPPN-Compiler (explained in the next section) and saved on the server. The browser-based drawing tool MethodDraw (http://codevisually.com/method-draw/) was extended towards this end, allowing users to define regularities between different primitive shapes. For example, Figure 2a shows how symmetries are designated and drawn via the automatic duplication of shapes along the y-axis (which can either be turned on or off). Additionally, the scalable vector graphic (SVG) output of MethodDraw was altered to comply with the standards of the CPPN-Compiler. Finally, the interface of MethodDraw was simplified and some features omitted, leaving only those features which are supported by the compiler; this design choice non-intrusively limits the users to only use shapes that can be meaningfully expressed via the genotypic representation.

At any point while drawing, the user can press the compile button on the web interface and send their current drawing (in SVG format) to the server back-end, where it is compiled into a CPPN. The compiled CPPN then seeds the initial population of images displayed back to the user via the Picbreeder applet on the front-end (Figure 2a). The user can evolve the images further by selecting one or more individuals to breed through the same interactive evolution method as in the original Picbreeder program [1].

In addition to the online drawing board, several features to support and motivate user interaction were implemented. The web application allows users to login to their personal account, to tag and give titles to their creations as well as to browse and rate the



(a) (b)

Fig. 2: **User Interface of DrawCompileEvolve.** (a) In the drawing interface the user can draw a SVG image on the canvas (left) and can compile and evolve it as a CPPN image (right) via the Picbreeder applet. (b) In the gallery interface the user can rate another user's CPPN image, see the original SVG image and evolve it further.

CPPN creations of other users (Figure 2b). Furthermore, by selecting an already saved CPPN image, any user can carry on evolving it and finally save it as another image. This gives users the opportunity to collaborate on evolving new images and CPPN structures.

5 CPPN-Compiler

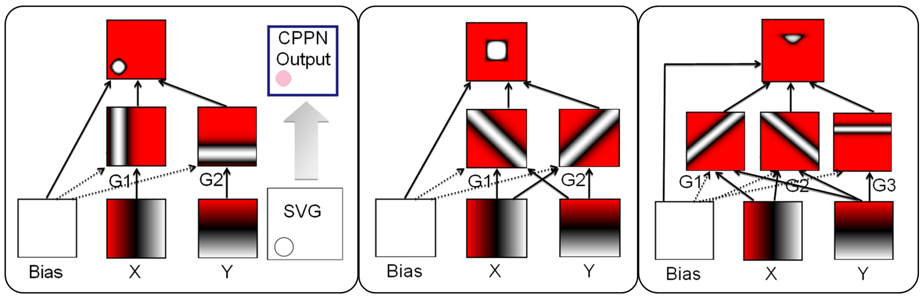
In the original Picbreeder, users typically start from randomly initialized populations (or elaborate on designs evolved by other users), in which case the CPPNs have randomized weights and a few randomly created hidden nodes. The NEAT algorithm then adds complexity over generations by adding new nodes and connections through mutations.

Instead of starting with random CPPNs, the approach presented in this paper enables users to jump-start evolutionary art by automatically converting their SVG drawings into CPPNs. The basic idea behind this *CPPN-Compiler* is that complex patterns with regularities can be represented by a few basic building blocks [14]. The current version of DrawCompileEvolve allows a set of primitive shapes to be converted to CPPN patterns, as well as arrange those in groupings such as symmetrical or repetitive patterns.

Importantly, defining these regularities allows coordinated and meaningful mutations on the generated structures. For example, because the compiled CPPN representation instills the fact that the two user-defined eyes in Figure 2a should be symmetric, one mutation can decrease the size of *both* eyes simultaneously. In effect, the CPPN-Compiler together with the DrawCompileEvolve tool allow the gradual transformation of the user's initial drawing into different artistic renderings.

5.1 Compiling Shape Primitives

The current version of the compiler allows the user to draw ellipses, rhombi, rectangles, triangles, and trapezoids. These shapes are then automatically converted into a CPPN representation, which is explained next:



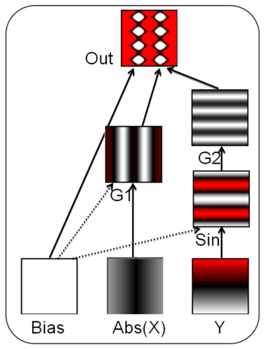
(a) Ellipsis (b) Rectangle (c) Triangle

Fig. 3: Example shape primitives and their CPPN representations.

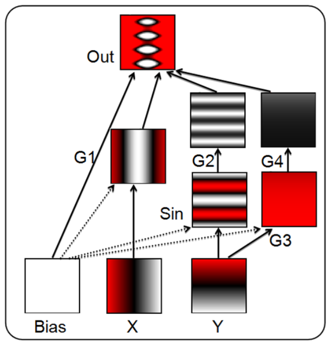
- **Ellipsis:** Constructing a CPPN that encodes a single ellipsis is achieved by using two nodes assigned with Gaussian activation functions ($G1$, $G2$) connected as shown in Figure 3a. The ellipsis' radii along the x and y axis are determined by the weights from the x-input to $G1$ and the y-input to $G2$ respectively. The position of an ellipsis is determined by the weight of the bias to $G1$ (horizontal movement) and $G2$ (vertical movement).
- **Rhombus:** A rhombus is constructed in the same way as an ellipsis, but using a different weight value on the connection between the bias and the output node.
- **Rectangle:** Constructing a rectangle is achieved via two nodes assigned with Gaussian activation functions ($G1$, $G2$) connected as shown in Figure 3b: unlike the rhombus, the x-input is also connected to $G2$ and the y-input to $G1$. The rectangle's width is determined by the weights from both x and y inputs to $G1$ and its height from both x and y inputs to $G2$. Positioning the rectangle is somewhat more involved, and requires that the x and y CPPN inputs are normalized in accordance to the rectangle's original dimensions and the size of the final image. Rotation has not yet been implemented for rectangles.
- **Triangle:** A triangle is constructed in a similar fashion as a rectangle, with the addition of a third Gaussian node ($G3$). For triangles with a rotation of 0^o or 180^o, the triangle's width is defined by the weights of links from x-input to $G1$ and $G2$ and its height by links from y-input to $G1$ and $G2$. $G1$ and $G2$ construct something akin to a rhombus, while the triangle is shaped via the connection between x-input and $G3$ (Figure 3c). Position of the triangle is defined by the links from the bias node to $G1$ and $G2$; when triangles are rotated at other angles (not 0^o or 180^o), position is also affected by the weight of the connection between bias and $G3$.
- **Trapezoid:** A trapezoid is constructed similar to a triangle by changing the weights of several connections, in order for $G3$ to "intersect" the patterns of $G1$ and $G2$ in a way that forms a trapezoid (at the correct angle and position) rather than a triangle.

5.2 Compiling Regularities

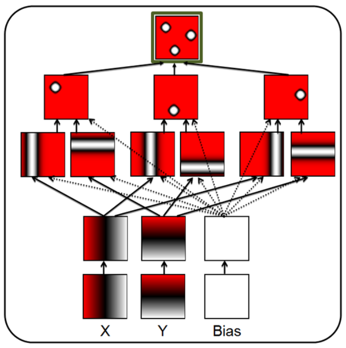
A user can assign a specific set of patterns on some of the shape primitives (the ellipsis and the rhombus at the time of writing). The current interface allows primitives to be



(a) Repetition with Symmetry



(b) Repetition with Variation



(c) Linking Multiple Patterns

Fig. 4: Compiling regularities into CPPN representations.

reflected along the vertical axis at the middle of the canvas (symmetry); alternatively the user can create multiple instances of a selected shape along the vertical axis (repetition), and optionally create a symmetrical array or have variation in the multiple instances.

- **Symmetry:** An important regularity in natural systems from human faces to animal bodies is symmetry. Symmetry along the y-axis can be compiled into a CPPN by applying an absolute function to the x coordinate before feeding it into $G1$.
- **Repetition:** Repetition of an ellipsis or rhombus can be compiled into a CPPN via a sine function. In the current version of DrawCompileEvolve, repetition along the y-axis can be encoded by applying a sine function on the y coordinate before it is input to $G2$ when drawing ellipses or rhombi. The frequency of the sine function determines how many copies of the primitive are shown on the canvas, and the distance between repeated primitives; the weight of the bias to the sine function determines the placement along the y axis of the sequence of primitives.
- **Repetition with symmetry:** The CPPN representation naturally allows the combination of repetition and symmetry by combining an absolute function (symmetry) and a sine function (repetition) in a manner depicted in Figure 4a.
- **Repetition with variation:** Repetition with variation is another fundamental regularity evident throughout nature (e.g. human fingers and toes, fern leaves, etc.). Repetition is compiled in the CPPN via the sine function; variation is introduced with the addition of two Gaussian nodes: $G3$ (using the y coordinate and the bias as inputs) and $G4$ (using the output of $G3$ as input). The weight of the connection between the bias node and $G3$ controls the variation of the output pattern, e.g. from small to large and vice versa.
- **Linking primitives:** Linking multiple primitives allows the user to group multiple pattern encodings into a single network. As the primitives initially share the weights from the same input nodes, mutations on the connections from the original input nodes may affect multiple primitives simultaneously [14]. As shown in Figure 4c, linking multiple primitives requires that the creation of a copy of all input nodes

(via an identity function), and the patterns' sizes and positions are normalized (in a similar way as individual ellipses of Figure 3a) before they are combined in the final output node.

6 Experiment

The web interface and persistent database of DrawCompileEvolve allows for any number of users to interact with the tool and evolve their own or others' creations. Creations included in this paper are collected primarily from early tests and internal trials. As the tool gains traction, it is expected that much more diverse and unexpected creations will be collected, and more branching from such images will occur by a growing community of users. As the web-based tool can be updated on-the-fly, future versions will likely include additions discussed in Section 8 to improve the quality of results.

Experimental Parameters: Because CPPN-NEAT differs from original NEAT only in its set of activation functions, it uses the same parameters [10]. Offspring had a 20% chance of link weight mutation, 10% chance of link addition, and 7% chance of node addition. The available CPPN activation functions were sigmoid, Gaussian, absolute value, cosine, and sine, all with equal probability of being added.

The compiled CPPNs (in their current state) have one brightness output. Because hue and saturation are set to fixed values (-2.0, -0.5), a different brightness value will produce different RGB colors. Currently, the color values of the original image are disregarded and instead chosen so that overlapping shapes have different colors, making them discernible.

7 Results

Figure 5 shows an indicative set of drawn SVG images which are compiled into CPPNs and displayed back to the user. The compiler seems to more or less faithfully represent the original shapes' position, rotation and outline, and maintains symmetry (e.g. the third image in Figure 5). The corners of triangles and rectangles appear softer in the CPPN image, which is not surprising given the smooth activation functions used (e.g. Gaussian and sine functions). Additionally, the proportions of shapes in the CPPN encoded images can sometimes vary slightly from the original image. These discrepancies are due to the CPPN's non-linear mapping from input to output values (which could potentially be compensated for in the future).

Acting as a seed for interactive evolution, the CPPN versions of user drawings can quickly generate interesting alternatives to the original drawing via interactive evolution. For instance, within five generations the duck in Figure 6 is transformed into a chicken. There is an interesting underlying reason for the transformational power of DrawCompileEvolve: unlike Picbreeder creations evolved "from scratch", users of DrawCompileEvolve ascribe specific semantic attributes to the otherwise abstract shapes they are drawing. In the case of Figure 6, the light gray triangle is a beak while the darker triangle is a foot (despite both shapes being of the same shape and size). After five generations, some of the shapes have new semantic attributes: the beak is still there,

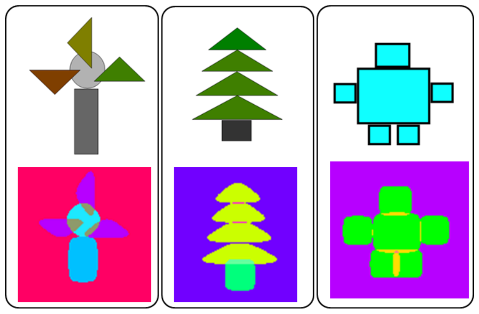


Fig. 5: Indicative user-created SVG images and corresponding CPPN images.

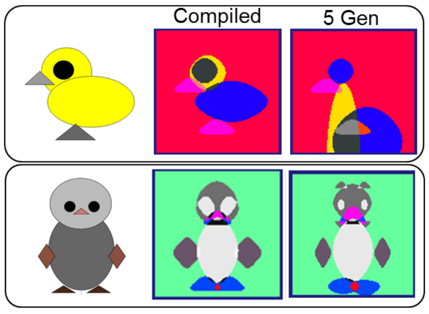


Fig. 6: Starting from a user's drawing, a chicken is evolved from a duck, and an ant from a penguin in five generations.

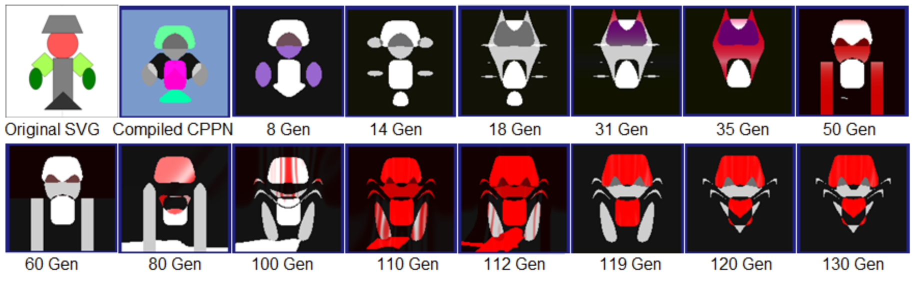


Fig. 7: When symmetry is maintained, one life-like artifact is transformed into another.

but the foot has been converted into a wing. This transformation leads to diagrammatic lateral thinking, enhancing the users' creativity as they evolve their creations [15].

Interactive evolution can fundamentally change the user's image by moving, transforming and recoloring existing shapes and by adding new shapes and background elements. While the user is in control of the process, it was observed that users tend to prefer to evolve the more radically different images among those presented in Picbreeder. Some additional criteria for user selection were often (a) whether the evolved image represented real-world objects (e.g. a chicken or a face) and (b) whether it retained the semantic attributes of the original image (e.g. whether the evolved image of Figure 6 still had a beak, or represented an animal).

While CPPNs can evolve into more complex and colorful images, some of the patterns were more persistent across generations of interactive evolution than others. Specifically, the symmetry compiled into the CPPN via the "symmetry" option of the drawing interface (see Figure 2a) was maintained even after more than 100 generations (e.g. in Figure 7). Although less pronounced, repetition (including repetition with symmetry as in Figure 8 and repetition with variation as in Figure 9) often persisted for many generations. There are likely two reasons for the persistence of these two features (as opposed to the individual ellipses in cases such as Figure 6). It is likely that the CPPN-Compiler itself biases this behavior by directly connecting the absolute function (for symmetry) and the sine functions (for repetition) to the input nodes, thereby re-

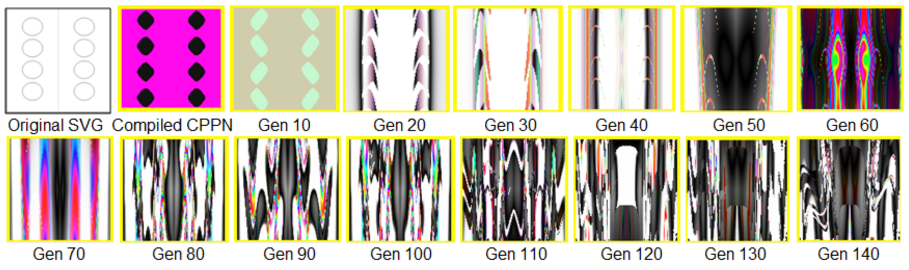


Fig. 8: Repetition with symmetry (especially symmetry) is maintained from the user's drawing (first image) and throughout IEC after more than 100 generations.

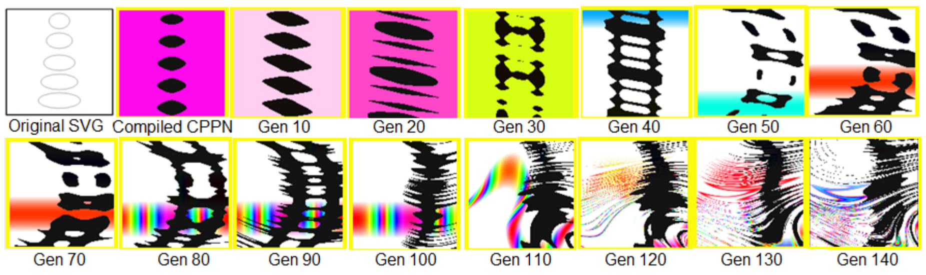


Fig. 9: Repetition with variation is maintained from the user's drawing (first image) and for 30 generations of IEC, before the images diverge to less obvious repetitive patterns.

ducing the likelihood of node additions through NEAT which would disrupt symmetry. A secondary reason is that symmetry is likely to be preferred among asymmetrical alternatives in the interactive evolution of Picbreeder, due to the human cognitive bias towards symmetry — especially the vertical symmetry used by the CPPN-Compiler.

The ability of vertical symmetry to create life-like visual artifacts is evidenced in Figure 7. The initial user drawing represented a human body with the necessary vertical symmetry. While the legs promptly disappear by the 8th generation, the vertical symmetry remains and eventually the CPPN image evolves into the face of a cat at generations 31-35. Further evolution retains vertical symmetry (although generations 80-112 include an asymmetrical feature at the bottom) and the CPPN image transforms into a gasmask head (at generation 60) and eventually into a spider at generation 130.

8 Discussion and Future Work

This study presented the first steps towards creating a mixed-initiative tool that allows users to automatically convert their vector drawings into evolvable representations, which can then be transformed through interactive evolution. There are several extensions to DrawCompileEvolve that would increase its expressivity and its ability to seed interactive evolution with desirable artifacts. Firstly, the CPPN-compiler could be enhanced to represent symmetry, repetition and related patterns along any axis (vertical, horizontal or diagonal). Due to the normalization of input when positioning rectangles, triangles and trapezoids within the CPPN-Compiler, groupings of Section 5.2 cannot work with those primitives; future work should address this limitation. Moreover, the accuracy of the compiler when dealing with multiple patterns and shapes could be enhanced, as linking primitives is less likely to represent the grouped shapes precisely. Finally, an important addition to DrawCompileEvolve is the carryover of color from SVG image to CPPN image: allowing the user to specify the color of each shape in the CPPN-Compiler would greatly enhance the user's creative control over the initial artifacts of the Picbreeder applet.

Additionally, in the future it will be important to more closely compare the artifacts evolved through a seeded approach with artifacts that are evolved from scratch. Seeding the initial population with meaningful images creates an evolutionary bias towards certain parts of the search space. While results in this paper show that it is possible to transform one life-like artifact into another (Fig. 7), an important question is how far these transformation can go and if a longer running interaction with DrawCompile-Evolve can generate the same breadth of image motifs as a system like Picbreeder [1].

Beyond two-dimensional drawings and evolutionary art, however, the principle of compiling user input into CPPNs can be applied to a variety of domains. Using the same drawing tool and treating the image as a heightmap, game terrain can be drawn by a human designer and (interactively) evolved by a computational designer in a similar manner to Sentient World [13]. The user can draw elliptical lakes (transformed into areas of very low altitude on the 3D terrain) or a repetitive pattern of mountains (transformed from rhombi into high altitude points of the 3D terrain); potentially, the user could also "paint" more than height values on the map, such as climate ranges, vegetation types or temperature maps. Other variants of DrawCompileEvolve can be used for the mixed-initiative design of agent behaviors: the CPPN image can represent the mapping between a range of sensors of a robot controller (input) and its angle and velocity (output). Since such a mapping is less intuitive than a purely visual imagistic representation, the users can observe the resulting simulated robot behaviors in real-time, providing insights into the effects of different changes in the mapping in a similar fashion to BrainCrafter [16].

9 Conclusion

This paper presented DrawCompileEvolve, a mixed-initiative drawing tool which allows users to seed interactive evolution with primitives and patterns that they find desirable. The drawing interface allows users to define a meaningful, visually appealing and personalized inspiration piece; this inspiration piece can then be further refined or transformed via the Picbreeder interactive evolutionary interface. Transforming vector art to CPPN images is accomplished via a CPPN-Compiler [14], which has been significantly refined and enhanced within DrawCompileEvolve. Early results have demonstrated the expressivity of CPPNs when directly controlled by human artists, as well as the power of interactive evolution to transform one semantically meaningful image, provided by the human user, into another equally meaningful image created by the algorithm.

Acknowledgments

We would like to thank the users of DrawCompileEvolve for their contributions. The research was supported, in part, by the FP7 ICT project C2Learn (project no: 318480) and by the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

1. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation (2011)
2. Clune, J., Lipson, H.: Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In: Proceedings of the European Conference on Artificial Life. (2011)
3. Risi, S., Lehman, J., D'Ambrosio, D., Hall, R., Stanley, K.O.: Introducing a marketplace for evolved content in the petalz social video game. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. (2012)
4. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3) (2012) 213–228
5. Hastings, E., Guha, R., Stanley, K.O.: NEAT particles: Design, representation, and animation of particle system effects. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games. (2007)
6. Hoover, A., Szerlip, P., Stanley, K.O.: Generating muscial accompaniment through functional scaffolding. In: Proccedings of the Sound and Music Computing Conference. (2011)
7. Hoover, A., Rosario, M.P., Stanley, K.O.: Scaffolding for interactively evolving novel drum tracks for existing songs. In: Proceedings of the European Workshop on Evolutionary and Biologically Inspired Music, Sound, Art and Design. (2008)
8. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE **89**(9) (2001) 1275–1296
9. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines **8**(2) (2007) 131–162
10. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2) (2002) 99–127
11. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Adaptive game level creation through rank-based interactive evolution. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG). (2013)
12. Clune, J., Chen, A., Lipson, H.: Upload any object and evolve it: Injecting complex geometric patterns into CPPNs for further evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation. (2013)
13. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient world: Human-based procedural cartography. In: Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt). Volume 7834, LNCS., Springer (2013) 180–191
14. Risi, S.: A compiler for CPPNs: Transforming phenotypic descriptions into genotypic representations. In: Proceedings of the AAAI Fall Symposium Series. (2013)
15. Yannakakis, G.N., Liapis, A., Alexopoulos, C.: Mixed-initiative co-creativity. In: Proceedings of the 9th Conference on the Foundations of Digital Games. (2014)
16. Risi, S., Zhang, J., Taarnby, R., Greve, P., Piskur, J., Liapis, A., Togelius, J.: The case for a mixed-initiative collaborative neuroevolution approach. In: Proceedings of the ALIFE Workshop on Artificial Life and the Web. (2014)