Sword Blade Generation based on CPPN

Mikhail Baryshev
G+PM Master's Student
The University of California Santa Cruz
mbaryshe@ucsc.edu
Supervisor: Jim Whitehead

ABSTRACT
In this paper, CppnSword is described, a tool for generating 2d pixel blades of swords based on human-driven CPPN-NEAT.

General Terms
Algorithms, Design, Experimentation.

Keywords
PCG, CPPN-NEAT, videogame, sword.

1. INTRODUCTION
We all experience the boom in Procedural Content Generation research and its applications. CPPNs are already used for 2d picture generation, like in a website Picbreeder [1]. As for games, CPPNs were used in Galactic Arms Race [2] in order to generate paths for projectiles. I wanted to combine those two and to make a CPPN-based generator for game assets, swords, for instance. There is a shortage of such sword generators in the field. Swords are just a step as CPPN-based generators might assist human designers in designing many in-game assets, especially for role-playing games.

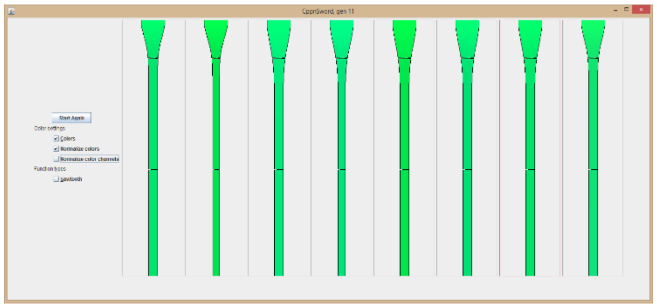
2. SYSTEM DESCRIPTION
2.1 System Overview
The system, made in Java, consists of user interface for driving the evolution and the underlining logic that keeps the CPPN and does mutations and crossover.

2.2 User Interface
The program's interface provides the user with eight possible sword designs, from with one is chosen and serve as a parent for further generation. (Seven swords are generated and the parent is kept on its place.) Those designs are set to fit in the space provided for them.

In addition, the user can toggle color options between "no color", "truncated colors", "normalization based on one color" (default) and "normalization of all colors." There is also a button to start the evolution over and a selection whether to include sawtooth function (off by default).

The text is used to help the user to operation the program, including the description on how to "lock" a sword design so it is not replaced (represented by a black square in top left corner) and how to case a crossover (clicks cause mutations by default).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or evil or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.



Color settings
[✓] Colors
[✓] Normalize colors
[ ] Normalize color channels
Function types
[ ] sawtooth

Figure 1. UI of CppnSword.

2.3 CPPN Representation
The system stores digraphs for CPPN representation, including "input", "output" and "inner nodes". The input nodes are used as input to the system; they are represented by a float parameter [0;1] that surveys the parameters of the sword among the blade length and a bias parameter, equal to 1. The output nodes provide width and 3 color channels (RGB) for each value of the input; that is, for each point along the sword length. The inner nodes contain mathematical functions including Sine, Gaussian and Sawtooth (optional) functions. The nodes are connected with each other using edges that have weight and that serve as multiplications.

CppnSword surveys the network with a set of inputs to get the output values (width and RGB) for each pixel of height of the image. The input values are propagated through the network to get the output values by applying the functions in the nodes and multiplying by the weight of the edges. When there are several inputs to an inner of output nodes, the values are added up.

The swords are mirrored among the y-axis to generate good-looking symmetric swords.

By default, the parameter input is connected to an inner node with a random function that is connected to the width output.

2.4 NEAT Mutation
When a user selects a sword to make a new generation, the topology is mutated. If follows the following procedure:

1. A new inner node with an empty function is made.

2. New edges are added, with decaying probability of ½ adding 1 edge, of ¼ of adding 2 edges and so on. Cycle check is performed.

3. Each edge is mutated with probability of ½ by multiplying it by a random value from Gaussian function with mean 1.

4. If the new node has no edges connected, it is discarded.

2.5 Crossover

The system is capable of doing one-point crossover between two swords. The point of crossover is selected to a random point of closest local width. Then, basically, a switch is introduced that selects a network given whether the enquiry point is above or below the point of crossover.

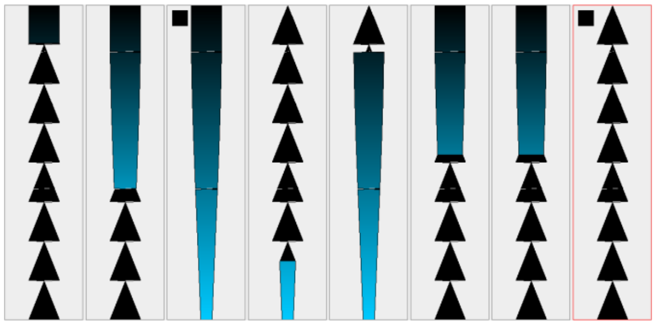


Figure 2. Result of a crossover between swords 3 and 8. Note that the "trees" are a result of Sawtooth function.

3. EVALUATION

The system is capable of generating objects that looks like swords. A cyclical nature of sword generation was observed. The generation was also testes with additional functions. The differences of visual output due to the color setting was aesthetically measured.

3.1 The Cyclical Nature of Sword Generation

CppnSword aretfacts look like sword in the first generations. Then, the artifacts stop resembling swords and lean towards some abstract figures. If continuing the generation, however, the objects start to resemble swords again when the width start to be extremely volatile.

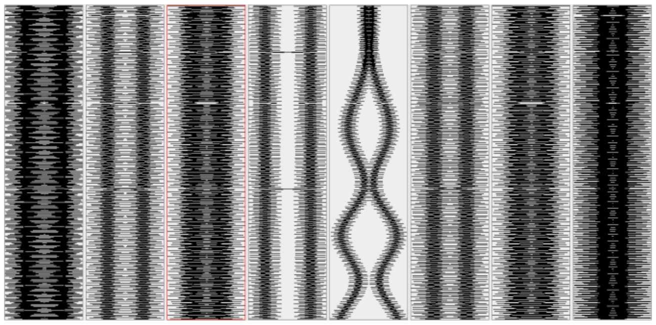


Figure 3. An emergence of an "Aftersword". Generation 41.

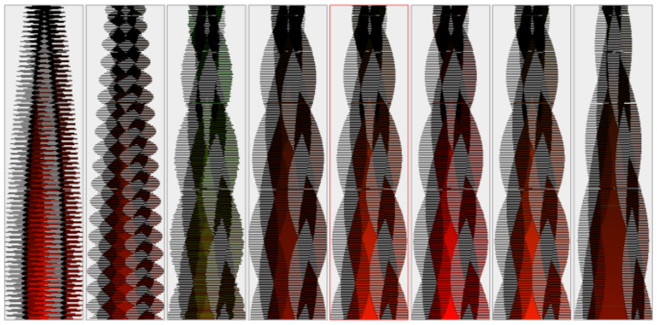


Figure 4. The "Afterswords" – shapes that resemble swords again. Generation 57.

3.2 Function Type Effects

CppnSword used Sine and Gaussian functions as defaults. An attempt to add a Sawtooth function was made, but the designs tend to converge unlike the pure Sine-Gaussian approach. However, the inclusion tend to generate some interesting designs that is why it is left in the interface, but off by default.

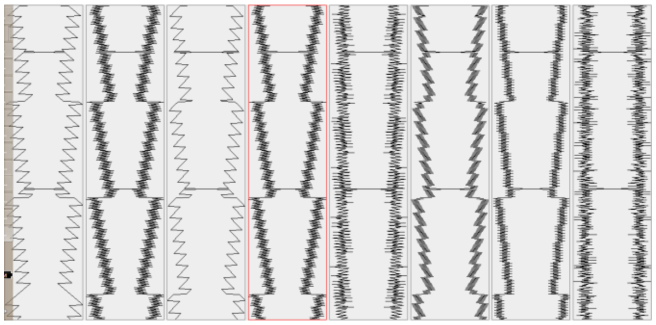


Figure 5. Sawtooth-generated shapes. Generation 86.

3.3 Color Settings Effect

Originally, the colors were truncated to the domain of [0;1]; given that the network provided a wide range of output values, the color was converging to teal or white pretty fast and had no opportunity to change again. Thus, color normalization was imposed; that is, all the colors are scaled so the largest color channel value among the sword is 1.

Then, the option to scale each individual color channel to fit in 1 was introduced. The swords, however, started to look over-saturated as several channels were able to get to 1 simultaneously. Thus, more "realistic" normalization based only on one channel is a default so the other color channels never reach 1; the result is darker and more diverse colors.

It worth noting that sometimes the swords are not symmetric in terms of color being on top of the sword's black shape; being a bug at first, it was left in the program, as the resulting effect is aesthetically pleasing.

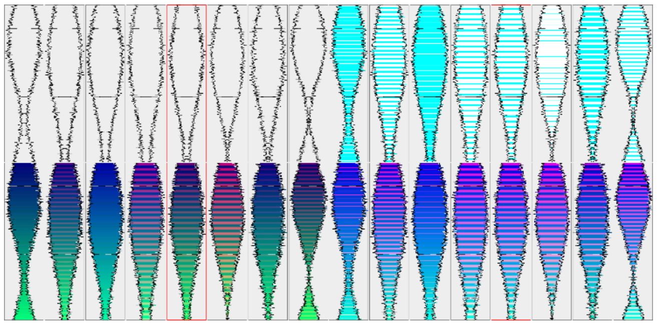


Figure 4. Different color settings. From left to right, top-down: no color, truncated color, 1 normalized, all normalized.

4. FUTURE WORK

CppnSword is not well-suited for use in content generation now as it lacks the functionality to export the sword image or save the topology. However, it was developed more as a concept and needs to be upgraded in the terms of content generation at first.

Other function might be used in the network and the change of the parameters of the generations might be useful to generate better sword-like content.

Crossovers of swords are an unexplored territory; the current version of the program is not powerful enough in that regard. There could be different crossover types, such as superimposing one sword on another or cross-fading widths and colors.

Moreover, more properties of the swords might be used; for instance, a curvature of the blade might be generated and in-game properties, such as damage per hit and magic enchantment, might be made based on the visual/color appearance of the sword.

A shift from swords to other types of content might be made as well.

5. CONCLUSION

CppnSword is a tool that is capable of generating swords and sword-like shapes. Given a relatively simple internal structure, the generative capabilities of the program are impressive.

I hope we will have more tools to assist game designers in content creation and, perhaps, that CPPNs are a step in the right direction.

6. ACKNOWLEDGMENTS

I would like to thank everyone who participated in testing of the software and Jim Whitehead, my supervisor.

7. CPPNSWORD

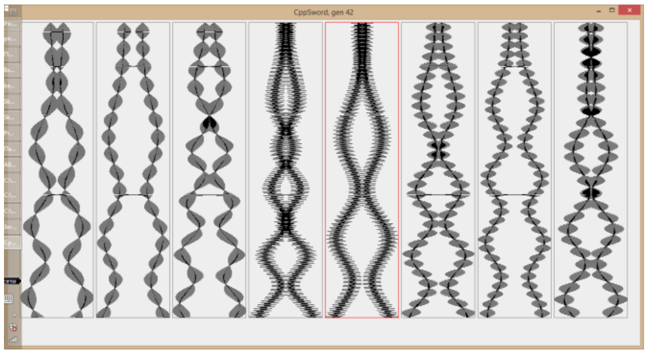
The source code for the project available at https://github.com/zemike/CppnSword and a compiled version is available at http://mbayshev.com/cppnsword.

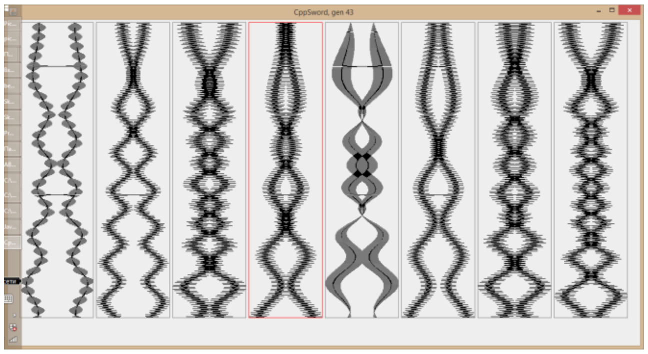
8. REFERENCES

[1] Picbreeder. [http://picbreeder.org/]

[2] Hastings, E.J.; Guha, R.K.; Stanley, K.O., Evolving content in the Galactic Arms Race video game, *Computational Intelligence and Games*, 2009, 241-248. DOI: http://dx.doi.org/10.1109/CIG.2009.5286468.

9. APPENDIX: SAMPLE OUTPUTS


CppSword, gen 42


CppSword, gen 43