Using Grammatical Evolution to Parameterise Interactive 3D Image Generation

Miguel Nicolau
Dan Costelloe
Natural Computing Research & Applications Group
University College Dublin
Dublin, Ireland
miguel.nicolau@ucd.ie, dan.costelloe@gmail.com

Abstract

This paper describes an Interactive Evolutionary system for generating pleasing 3D images using a combination of Grammatical Evolution and Jenn3d, a freely available visualiser of Cayley graphs of finite Coxeter groups. Using interactive GE with some novel enhancements, the parameter space of the Jenn3d image-generating system is navigated by the user, permitting the creation of realistic, unique and award winning images in just a few generations. One of the evolved images has been selected to illustrate the proceedings of the EvoStar conference in 2011.

1 Introduction

Evolutionary Algorithms permit the creation of candidate solutions from arbitrarily coarse- or fine-grained representations. In an Interactive Evolutionary Design context (i.e. with human evaluators playing the part of the fitness function), user-fatigue is more likely to occur before acceptable quality solutions emerge when the genotype representation is too fine-grained, due to a long, repetitive process with no immediately pleasing results.

for example, consider a toy Evolutionary image generation task where the goal is to produce a simple black and white image on an $N \times N$ grid. The image could be encoded as a binary string of length N^2 – a representation that fits well with a Genetic Algorithm implementation.

This simple representation is powerful in that it permits the construction of *every* possible 2-dimensional monochrome image for a given value of N. If this were a standard, non-interactive optimisation problem, this kind of representation would probably be suitable, since the space of all possible solutions is covered, meaning that with the right conditions, high-quality solutions are almost guaranteed as output from evolution.

But in an interactive setting, this type of (fine-grained) representation makes the construction of even the most basic shapes on the canvas a slow and difficult process. Adding to the difficulty is the potentially destructive nature of the genetic operators of the GA. For this type of problem, the use of pre-defined building blocks (e.g. lines, rectangles, curves) is more likely to produce pleasing or interesting images in a shorter amount of time, while reducing user-fatigue. This notion of building-block creation and re-use has been employed in other artistically focused evolutionary systems such as *Picbreeder* [17].

The approach taken in this work was somewhat different than that of typical evolutionary art approaches. In this case, the evolutionary algorithm is not actually constructing the image, but rather parametrising a 3-dimensional visualiser of complex mathematical structures.

There were two main reasons for this approach. The first was speed of development; by using a freely available tool that generates the images, there was no graphical development involved, so all that was required was the integration of the evolutionary algorithm with the visualiser tool, evolving the parameters for the visualiser. The second reason was a shorter evolutionary process. The graphical visualiser used generates fully constructed images, which are quite pleasing to the eye[1]; the time required for the interactive evaluation process to reach images that fulfill the objective is therefore potentially much shorter.

The results obtained are certainly unique; not only that, but they were also obtained with very short runs of the system, thus complying with the objectives stated above. Finally, they are visually appealing, not only for the authors, but also for different communities: one of the evolved images won the EvoStar 2010 art competition, and now illustrates the proceedings cover, while another won a competition to be chosen as the logo for a research cluster, located in the University College Dublin, Ireland.

This paper describes the implementation of this work. It presents the evolutionary algorithm used in Section 2, followed by a brief introduction to Coxeter Matrices and the visualiser used, in Section 3, and finally describes the experiments conducted, in Section 4. Section 5 then draws some conclusions.

2 Grammatical Evolution

Grammatical Evolution (GE) [12, 15] is an evolutionary approach that specifies the syntax of possible solutions through a context-free grammar, which is then used to map binary strings onto syntactically correct solutions. Those binary strings can therefore be evolved by any search algorithm; typically, a variable-length genetic algorithm is used.

The use of a grammar to specify the syntax of solutions allows the application of GE to a variety of domains; these are as diverse as horse gait optimisation [8], wall shear stress analysis in grafted arteries [2], and optimisation of controllers

[1] Although a subjective statement, this opinion was also shared by enough voters to deem one of the images the winner in a competition against other images produced by evolutionary means.

```
<Pic>       ::= <Pic> <Term>
              | <Term>
<Term>      ::= <Var>
              | <Op> <Term>
<Op>        ::= Grow(<Value>)
              | Shrink(<Value>)
              | Rotate(<Value>)
<Var>       ::= square
              | circle
              | triangle
<Value>     ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figure 1: Example grammar for defining simple shapes.

for video-games [3]. This also includes earlier applications to evolving art, such as the evolution of logos using Lindenmayer systems [11], musical scores [13], generation of digital surfaces [4], and architectural design [18, 7].

2.1 Example Mapping Process

To illustrate the mapping process, consider the (simplified) shape grammar shown in Fig. 1. Given an integer (genotype) string, such as `(4, 5, 4, 6, 7, 5, 9)`, a program (phenotype) can be constructed, which respects the syntax specified in the grammar.

This works by using each integer to choose productions from the grammar. In this example, the first integer chooses one of the two productions of the start symbol `<Pic>`, through the formula $4\%2 = 0$, i.e. the first production is chosen, so the mapping string becomes `<Pic> <Term>`.

The following integer is then used with the first unmapped symbol in the mapping string, so through the formula $5\%2 = 1$ the symbol `<Pic>` is replaced by `<Term>`, and thus the mapping string becomes `<Term><Term>`.

The mapping process continues in this fashion, so in the next step the mapping string becomes `<Var> <Term>` through the formula $4\%2 = 0$, and through $6\%3 = 0$ it becomes `square <Term>`. Continuing in this fashion, all non-terminal symbols in the growing expression are mapped, until the final program becomes `square Rotate(9)`, which can then be used to generate a shape.

Sometimes the integer string may not have enough values to fully map a syntactic valid program; several options are available, such as reusing the same integers (in a process called wrapping[12]), assigning the individual the worst possible fitness, or replacing it with a legal individual. In this study, an unmapped individual is replaced by its parent.

3 Cayley Graphs

Elementary mathematical group theory teaches us that a group is a special type of set, combined with a number of operations that obey fundamental algebraic rules. For the visually inclined, a Cayley graph permits a diagrammatic representation of the structure of a group with respect to a generating subset. For

a given discrete group, G, altering the generating set S can produce visually interesting geometric consequences for the Cayley graph representation of G. An example of this is shown in Fig. 2.

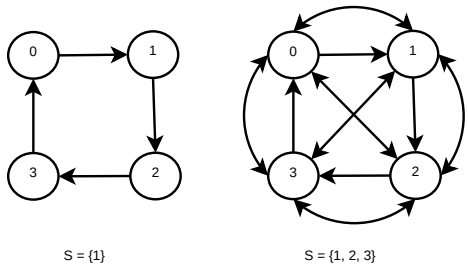

Figure 2: Two Cayley graphs of the cyclic group $Z/4Z$ produced by two different generating sets.

Increasing the complexity of these graphs and corresponding generating sets can in fact generate interesting, visually appealing structures. A complete description of the underlying mathematics at work to create and visualise such graphs is outside of the scope of this paper, as this would require sections on Coxeter Groups, Reflection Groups, discrete groups, topologies and many other aspects of group theory[2]. There are, however, available tools for the visualisation of such graphs; the Jenn3d system is one of them.

3.1 Jenn3d

Jenn3d [10] is a freely available tool developed by Fritz Obermeyer, which generates visualisations of Cayley graphs of finite Coxeter matrices; it does so using the Todd-Coxeter algorithm, and projects them onto Euclidean 3D space, by embedding them into a 3-sphere.

It is a very simple to use, yet powerful piece of software; its generating parameters are: the Coxeter matrix; the sub-group generators; a set of vertex stabilising generators; specific definitions of edges; specific definitions of faces; and a set of vertex weights.

Thanks to the ability of Grammatical Evolution to evolve parameters to the Jenn3d system that *just work*, it is not necessary to delve into these concepts in any great detail since their complexity can be abstracted away into an image-generating black box. This is obviously very convenient, however we feel that it is also quite a valuable contribution of this work – we as EC researchers and practitioners *do not need to know* the inner complexity of the black box, be it Jenn3d or otherwise. All that is needed is the means to navigate the search space of black box parameters, guiding the system to the generation of visually pleasing images.

[2]Many books are available on the subject, a good resource is Holt *et al* [6].

4 Experiments

4.1 Setup

A grammar was designed for GE, specifying the exact syntax of the required (and optional) parameters for Jenn3d. Some were tricky to encode; many parameter combinations make Jenn3d crash, or just get stuck in an endless computational loop. The solution to this was to use GE as the main executable, and make external calls to Jenn3d for each evaluation process; if the external call fails, a fitness of 0 (i.e. the worst fitness) is given to that set of parameters.

If the call is successful, the 3D visualiser appears on the screen; the user can then interact with the image, examining its 3-dimensional structure. If a particular viewing angle is found, Jenn3d has options to save a snapshot onto file; additional options were encoded onto Jenn3d for this work: a way to save the parameter combination onto a "best-parameters" file, and a scoring method. Fig. 3 illustrates this.

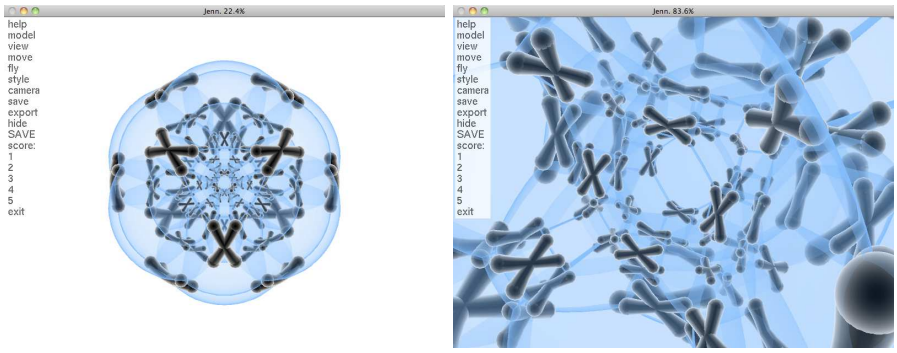

Figure 3: The Jenn3d interface, along with the extensions encoded. An example image is shown in the left; the same image, when zoomed in and rotated (right), can produce a dramatically different view angle.

The scoring options are the values 1 to 5. If the user gives the score 1 to a structure, it will be replaced by a random structure at the next generation; likewise, due to the 20% elitism replacement used, the structures with the maximum score per generation are guaranteed to remain in the next generation. This ensures that the user retains a degree of control over the generational mechanics, while allowing the evolutionary process to proceed as normal.

Additional novel approaches were used in this work. First, the size of the initial population was larger, to present an initial large spectrum of structures to the user; this size is then culled after the first generation, to the population size chosen for the evolutionary algorithm.

Another novel approach was the encoding of crossover points in the grammar. This is a technique presented recently for GE [9], in which a specific symbol is used in the grammar, to label crossover points; the evolutionary algorithm

then only slices an individual according to these points. This technique was particularly useful for the work presented: many of the parameters passed to Jenn3d specify styling options, which can therefore be exchanged as a whole between different individuals (a 2-point crossover operator was used). The same crossover points are used in both individuals, that is, vertex weights will only be exchanged with vertex weights; the length of the exchanged blocks, however, is variable. This makes crossover act mainly as an exploitation operator; as the exchange of parameters can dramatically change the visual appearance of a coxeter matrix, it made sense to limit the role of crossover to the exploration of such combinations. A standard point mutation operation is also used, which ensures the creation of novel parameter values. Fig. 4 shows a section of the used grammar, along with the encoded crossover points.

```
<cmdline> ::= ./jenn <GEXOMarker> -c <CoxeterMatrix> <GEXOMarker>
              <StabilizingGenerators> <GEXOMarker> <Edges> <GEXOMarker>
              <Faces> <GEXOMarker> <VertexWeights> <GEXOMarker>
<CoxeterMatrix>::= <Torus> | <FreePolyhedra> | <FreePolytope>
<StabilizingGenerators> ::= "" | -v <Comb0123>
<Edges>          ::= "" | -e <EdgeSet>
<Faces>          ::= "" | -f <FaceSet>
<VertexWeights> ::= "" | -w <Int1-12> <Int1-12> <Int1-12> <Int1-12>
```

Figure 4: Section of the grammar used; crossover points are encoded using the special non-terminal symbol `<GEXOMarker>`.

The experimental parameters used are shown in Table 1. To ensure all individuals in the initial population were valid, a form of population initialisation [14] was used. Also, the mutation rate was set such that, on average, one mutation event occurs per individual (its probability is dependent on the length of each individual). Finally, note that there is no maximum number of generations; evolution will always continue, until the user decides to terminate the execution.

Table 1: Experimental Setup

Initial Population Size	20
Evolutionary Population Size	10
Derivation-tree Depth (for initialisation)	10
Selection Tournament Size	10%
Elitism (for generational replacement)	20%
Crossover Ratio	50%
Average Mutation Events per Individual	1

4.2 Results

A typical run of the system seems to give many undesirable images on the first generation; some cause Jenn3d to crash, while others are visually displeasing.

After the first generation, however, the system seems to settle onto a sequence of very pleasing images, based on variations of the initial best images.

There is always novelty being introduced into the population, and the user has an active part on this process, by attributing a fitness score of 1 to displeasing structures, which forces these to be replaced by novel ones. A single run of the system can therefore generate a wide variety of images; once a user has explored many variations of a style, he/she can start attributing them fitness scores of 1, which effectively purges the population of these structures, and ensures that new, unseen structures are present in the next generation.

Fig. 5 shows examples of different variations of a pleasing image, obtained mostly through the exploration power of the new crossover operator. Fig. 6, on the other hand, shows many different images extracted from a single run of the system, a result achievable through the user-control technique explained. This is possible both due to the variety of structures which Jenn3d can project, and also to the exploration of the parameter space by GE. Note that all these figures are shown zoomed out; rotation, zooming and *fly-in* transformations can seriously alter the style achieved. Finally, Fig. 7 shows the image that won the EvoStar 2010 Art Competition.

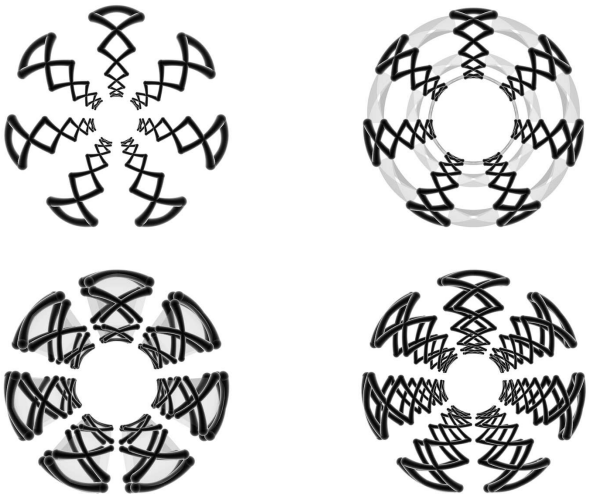

Figure 5: Example image and close variations achievable with the genetic operators used.

5 Conclusions

This paper has presented a novel application of Grammatical Evolution to Evolutionary Art that treats the task of producing pleasing Jenn3d / Coxeter visu-

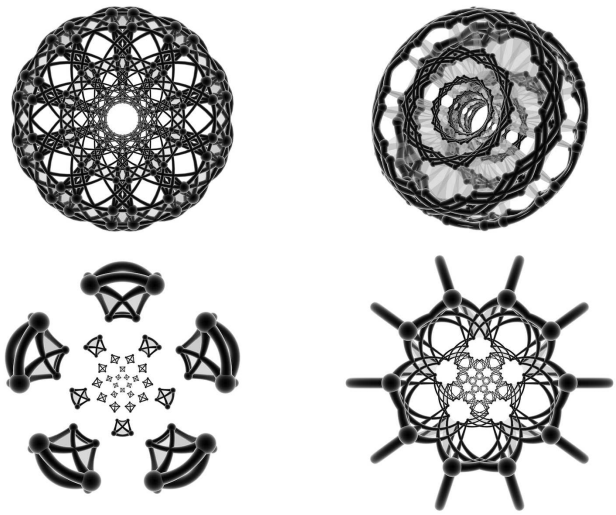

Figure 6: Example of the variety of structures achievable from a single run.

alisations as a parameter search problem. In fact, GE has been particularly well suited to this problem thanks to the construction of a grammar that encapsulates the complexity of the parameters of the image-generating Jenn3d system. The ease of use of both GE and Jenn3d made them easily combinable, which not only resulted in a fast implementation, but also allowed the generation of unique and very often pleasing images.

A fair criticism of the use of black-box such as Jenn3d is that the images produced will always be limited to the space of possibilities that the black-box is capable of creating. This is indisputable – doing so constrains the space of potential solutions and for certain problems this should be avoided in order to gain maximum coverage of the search space.

However, there are also cases where jump-starting is a sensible thing to do. In this case, the space of possible solutions is still sufficiently large that a spread of pleasing and not so pleasing images can come about. The necessity of interactive fitness assignment is just as much a requirement as it would be for a system producing arbitrary images. The advantage of using a system such as Jenn3d is that the user will not spend time in the early generations evolving the fundamentals.

The encoding of crossover points in the grammar also worked with great effect in this work. The crossover operator was originally designed [5] to work just like in nature, that is, to allow two chromosomes to exchange building-blocks; there has been a great dispute over the years, however, as to the real exploitation nature of crossover, and in fact to the existence of exchangeable building-blocks in Genetic Programming systems [1, 16]. In this work, they do exist, and the crossover operator was encoded to take full advantage of this fact.

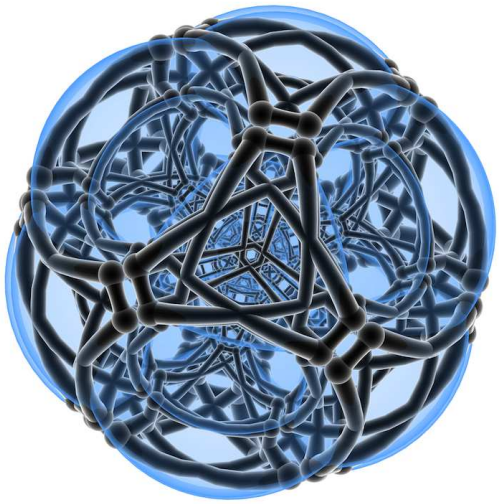

Figure 7: The winning image, elected to illustrate the 2011 EvoStar proceedings.

Finally, some of the images generated by this approach were submitted to a few competitions, with award-winning results: one has won the Evolutionary Art competition at EvoStar 2010, and another has been chosen as a logo representation for a research cluster in University College Dublin.

Acknowledgments

This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

[1] Angeline, P.J.: Subtree crossover: Building block engine or macromutation? In: et al., J.R.K. (ed.) Genetic Programming 1997: Second Annual Conference, Stanford, USA, July 13-16, 1997, Proceedings (1997)

[2] Azad, R.M.A., Ansari, A.R., Ryan, C., Walsh, M., McGloughlin, T.: An evolutionary approach to wall shear stress prediction in a grafted artery. Applied Soft Computing 4(2), 139–148 (2004)

[3] Galván-López, E., Swafford, J.M., ONeill, M., Brabazon, A.: Evolving a ms. pacman controller using grammatical evolution. In: et al, C.D.C.

(ed.) Proceedings of the EvoWorkshops 2010 on Applications of Evolutionary Computation, Istanbul, Turkey, April 7-9, 2010, Proceedings. Lecture Notes in Computer Science, vol. 6024, pp. 161–170. Springer (2010)

[4] Hemberg, M., O'Reilly, U.M.: Extending grammatical evolution to evolve digital surfaces with genr8. In: Keijzer, M., O'Reilly, U.M., Lucas, S.M., Costa, E., Soule, T. (eds.) Genetic Programming, 7th European Conference, EuroGP 2004, Coimbra, Portugal, April 5-7, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3003, pp. 299–308. Springer (2004)

[5] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)

[6] Holt, D.F., Eick, B., O'Brien, E.A.: Handbook of Computational Group Theory (Discrete Mathematics and Its Applications). Chapman and Hall/CRC (2005)

[7] McDermott, J., Griffith, N., O'Neill, M.: Interactive EC control of synthesized timbre. Evolutionary Computation 18(2), 277–303 (2010)

[8] Murphy, J.E., O'Neill, M., Carr, H.: Exploring grammatical evolution for horse gait optimisation. In: et al, M.G. (ed.) Genetic Programming, 12th European Conference, EuroGP 2009, Tübingen, Germany, April 15-17, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5484, pp. 579–584. Springer (2009)

[9] Nicolau, M., Dempsey, I.: Introducing grammar based extensions for grammatical evolution. In: IEEE Congress on Evolutionary Computation, CEC 2006, Vancouver, BC, Canada, July 16-21, 2006, Proceedings. pp. 2663–2670. IEEE Press (2006)

[10] Obermeyer, F.: Jenn3d for visualizing coxeter polytopes. http://www.math.cmu.edu/ fho/jenn/ (June 2010)

[11] O'Neill, M., Brabazon, A.: Evolving a logo design using lindenmayer systems, postscript & grammatical evolution. In: IEEE Congress on Evolutionary Computation, CEC 2008, Hong-Kong, June 1-6, 2008, Proceedings. pp. 3788–3794. IEEE Press (2008)

[12] O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, Genetic programming, vol. 4. Kluwer Academic Publishers (2003)

[13] Reddin, J., McDermott, J., Brabazon, A., O'Neill, M.: Elevated pitch: Automated grammatical evolution of short compositions. In: et al, M.G. (ed.) Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computation, Tübingen, Germany, April 15-17, 2009, Proceedings. Lecture Notes in Computer Science, vol. 5484, pp. 579–584. Springer (2009)

[14] Ryan, C., Azad, R.M.A.: Sensible initialisation in grammatical evolution. In: Barry, A.M. (ed.) GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference. pp. 142–145. AAAI, Chigaco (July 2003)

[15] Ryan, C., Collins, J.J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds.) First European Workshop on Genetic Programming 1998. pp. 83–95. Springer, Berlin (1998)

[16] Sastry, K., O'Reilly, U.M., Goldberg, D.E., Hill, D.: Building block supply in genetic programming. In: Riolo, R., Worzel, B. (eds.) Genetic Programming Theory and Practice, chap. 4, pp. 137–154. Kluwer Publishers, Boston, MA, USA (November 2003)

[17] Secretan, J., Beato, N., D Ambrosio, D.B., Rodriguez, A., Campbell, A., Stanley, K.O.: Picbreeder: evolving pictures collaboratively online. In: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems. pp. 1759–1768. CHI '08, ACM, New York, NY, USA (2008)

[18] Shao, J., McDermott, J., ONeill, M., Brabazon, A.: Jive: A generative, interactive, virtual, evolutionary music system. In: et al, C.D.C. (ed.) Proceedings of the EvoWorkshops 2010 on Applications of Evolutionary Computing, Istanbul, Turkey, April 7-9, 2010, Proceedings. Lecture Notes in Computer Science, vol. 6025, pp. 341–350. Springer (2010)