A Step Towards the Evolution of Visual Languages

Penousal Machado and Henrique Nunes

CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal machado@dei.uc.pt

Abstract. Traditional Evolutionary Art systems allow the evolution of individual artworks. We present a novel Evolutionary Art engine for the evolution of visual languages. In our approach, each individual is a context free grammar that specifies an entire family of shapes following the same production rules. Therefore, search takes place at an higher level of abstraction (families of shapes) than the one typically explored in Evolutionary Art systems (individual shapes). The description of the system gives particular emphasis to the novel aspects of the approach and to the generative potential of the representation.

1 Introduction

Stiny and Gips [1] introduced the concept of Shape Grammars, which: "are similar to phrase structure grammars, which were introduced by Chomsky in linguistics. Where phrase structure grammars are defined over an alphabet of symbols and generate one-dimensional strings of symbols, shape grammars are defined over an alphabet of shapes and generate n-dimensional shapes."[1]. Stiny and Gips have successfully built shape grammars that capture the "language" of Frank Lloyd Wright's prairie houses, Mughul Gardens, Palladian Plans, etc. Additionally, they were also able to use these grammars to produce new instances of the same language, e.g. to create new prairie house designs that obey Frank Lloyd Wright's style to the point of being indistinguishable, even to experts, from his original works [2].

Although these grammars are hand-built, and result from a complex process of analysis and formalization of the hidden rules followed in the originals, these results show that: (i) it is possible to capture specific visual languages using a set of production rules; (ii) it is then possible to use this set of rules to automatically generate new objects that belong to the same visual language.

The main motivation for the present work is the development of a system for the evolution of novel visual languages. For that purpose we created an evolutionary engine where each individual is a context free grammar, and developed appropriate genetic operators, including several mutation operators and graph-based crossover.

The use of the Context Free Design Grammar (CFDG) language [3] for representation allows the specification of complex families of shapes through a compact set of rules, and has several potential advantages over typical Evolutionary

Art (EA) representations. A brief overview of current evolutionary art systems, focusing on representation issues, is presented in Section 2 to allow a better contextualization of the present work, while, in Section 3, we describe the most relevant characteristics of CFDG. In the fourth Section we describe our evolutionary engine, and, in section 5, we present some of the experimental results attained. Finally, in Section 6, we draw some final conclusions and indicate future work.

2 Related Work

A thorough survey of EA systems beyond the scope of this paper (see, e.g., [4] for a in-depth survey). Here we present a brief overview, focusing on the issues that are of most relevance to the present work, namely, representation scheme and generation abilities of the system. The most popular EA approach is inspired in the seminal work of Karl Sims [5]. It uses Genetic Programming (GP) to evolve populations of images. Each genotype is a tree that encodes a LISP-like symbolic expression. The internal nodes of the tree are functions (typically arithmetic, trigonometric and image processing operations) and the leafs are terminals (typically x and y variables and random constants). The rendering of the expression results in a phenotype, e.g. an image. More often than not user-guided evolution is employed, i.e., the user assigns fitness to the images, thus indirectly determining the survival and mating probabilities of the individuals. The fittest individuals have a higher probability of being selected for the creation of the next population, which is generated through the recombination and mutation of the genetic code of the selected individuals.

The use of Genetic Algorithms (GAs) coupled with a fixed or variable length string representation is also frequent. In these cases the most common approach is *parametric* evolution [4]. In other words, the genotype encodes a set of parameters that determine the phenotype. Among other applications, this approach has been used to evolve cartoon faces, fractal shapes, fonts [4]. The use of EC approaches to the evolution of line-based drawings, 3D shapes, l-systems, filters, etc., has also been explored. Although the application area and implementation details vary, most systems can be seen as instances of *expression-based* or *parametric* evolution.

As Machado and Amílcar [6] point out, most *expression-based* EA systems are theoretically able to create any image (see also [7]). Nevertheless, in practice, the image space that is actually explored depends heavily on the particularities of the system (primitives, genetic operators, genotype-phenotype mapping, etc.). In other words, and notwithstanding works such as [8] that describes an approach to iterative stylistic change in EA, most systems have an identifiable *signature* that naturally emerges from the interactions between its different components. In parametric evolution models, system signature is even stronger since: "...creating a parametric model implicitly creates a set of possible designs or a solution space." [4] Thus, there are strong contraints that limit the search-space and define the type of imagery produced by the system.

Finally, to the best of our knowledge, there are two reported examples of the use of CFDG in the context of Evolutionary Art. Unfortunately, none of

them allows the evolution of visual languages. As the name indicates *CFDG Mutate* [9] only allows the application of mutation operators, which is limiting, and does not handle non-deterministic grammars, which means each individual represents a single shape (see Section 3). Saunders and Kazjon [10] present a parametric evolution model that evolves parameters of specific CFDG hand-built grammars. Although this allows some degree of exploration, in essence it has the same shortcomings as other parametric evolution approaches.

3 Context Free

Context Free [11] is a popular open-source application that renders images specified using a simple language entitled Context Free Design Grammar (CFDG). In essence, and although the notation is different from the one used in formal language theory, a CFDG program is a context free grammar, i.e. a 4-tuple: (V, Σ, R, S) where,

1. V is a set of non-terminal symbols
2. Σ is a set of terminal symbols
3. R is a set of production rules that map from V to $(V \quad \Sigma)$
4. S is the initial symbol

In Fig. 1 we present a simple grammar and the image generated by it. Programs are interpreted by starting with S (in this case $S = TREE$, as defined by the *startshape* directive) and proceeding the expansion of the production rules in breath-first fashion. Pre-defined V symbols call drawing primitives (e.g. CIRCLE draws a circle) while predefined Σ symbols produce semantic operations (e.g. *size* produces a scale change, y moves forward, etc.). Program interpretation is terminated when one of the two following criteria is met: (i) There are no V symbols left to expand; (ii) The further expansion does not change the image (E.g.: although the recursive loop of the grammar presented in Fig. 1 is endless, the set of transformation is *contractive* [12], after a few iterations we reach a size smaller than pixel size and, therefore, further expansion will not cause visible differences). This second termination criterium has no parallel in formal language theory, but is similar to the termination criterium used in Iterated Function Systems (IFSs) rendering.

The grammar depicted in Fig. 1 is deterministic, there is exactly one rule for each V symbol, therefore its interpretation will always result in the same image. To specify languages of shapes we have to resort to non-determinism. In Fig. 2 we present a non-deterministic version of this grammar, with two different production rules for the 'TREE' symbol. When several production rules may be applied one of them is selected randomly and the expansion of the grammar proceeds. One can control the relative probability of selection by specifying a weight after the V symbol[1] (in this case, 0.8 for the first rule and 0.2 for the second).

[1] The same effect can be attained by making copies of the production rule we wish to use more frequently. So this does not violate the formal language theory definition of a context free grammar.

```
startshape TREE
rule TREE {
     CIRCLE {}
     TREEA {size 0.95 y 1.6}}
rule TREEA {
     CIRCLE {}
     TREEB {size 0.95 y 1.6}}
rule TREEB {
     CIRCLE {}
     TREEC {size 0.95 y 1.6}}
rule TREEC {
     CIRCLE {}
     TREED {size 0.95 y 1.6}}
rule TREED {
     CIRCLE {}
     TREE {size 0.95 y 1.6 rotate 45}
     TREE {size 0.95 y 1.6 rotate -45}}
```

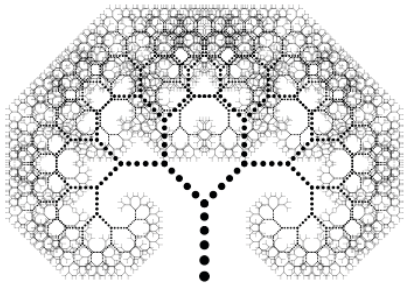

Fig. 1. A deterministic grammar and the tree-like shape generated by it.

```
startshape TREE

rule TREE 0.80 {
     CIRCLE {}
     TREE {size 0.95 y 1.6}
}


rule TREE 0.20 {
     CIRCLE {}
     TREE {size 0.95 y 1.6
            rotate 45}
     TREE {size 0.95 y 1.6
            rotate -45}}
```

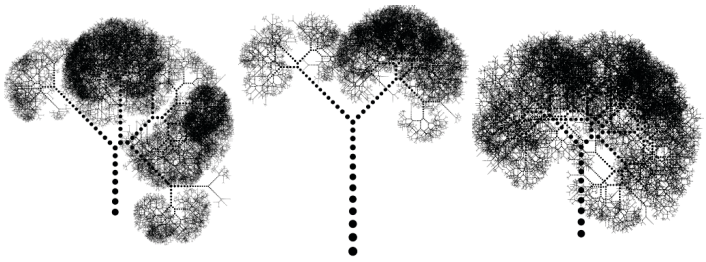

Fig. 2. A non-deterministic version of the grammar presented in Fig. 1 and instances of the family of tree-like shapes generated by it.

4 Evolutionary Context Free Art

In this section we describe our evolutionary engine. For the sake of parsimony we will avoid mentioning the implementation details and focus on the key components. An in-depth description is left for a future opportunity.

4.1 Representation

Each genotype is a well-constructed CFDG grammar. Internally the genotype is represented by a directed graph where each node encapsulates a production rule. For each node, N_i, outgoing edges are created in the following way:

1. Let V_i be the set of all V symbols that the production generates
2. Let M_i be the set of all nodes representing production rules that may be triggered by V_i symbols
3. Establish edges from N_i to all M_i nodes

For instance, the grammar of Fig. 1, results in the following edges: $TREE \rightarrow TREEA$, $TREEA \rightarrow TREEB$, $TREEB \rightarrow TREEC$, $TREEC \rightarrow TREED$;

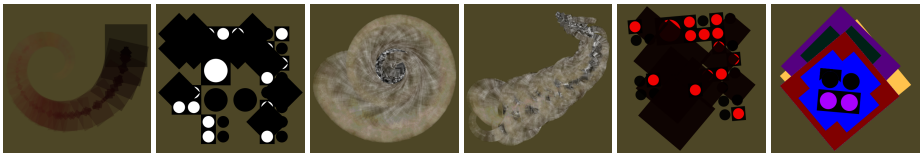

Fig. 3. The two leftmost images are the parents, the remaining ones are results of their crossover

for the grammar of Fig. 2 we would have $TREE_1$ $TREE_1$, $TREE_1$ $TREE_2$, $TREE_2$ $TREE_1$, $TREE_2$ $TREE_2$, where $TREE_1$ and $TREE_2$ represent the nodes that would be created for the first and second production, respectively.

The phenotype is rendered using Context Free. Infinite non-contractive loops may occur. To cope with this problem we specify a maximum amount of time for rendering. If that limit is reached rendering stops and the current image is considered the phenotype.

4.2 Genetic Operators

The design of genetic operators that are well-suited to the adopted representation is vital for the success of any evolutionary algorithm. In our case the biggest challenge was to design a recombination operator that allows the meaningful exchange of genetic material between individuals. Given the nature of the representation we developed a graph-based crossover operator based on the one presented by Pereira et al. [13]. In simple terms, this operator, inspired in the standard GP swap-tree crossover, allows the exchange of subgraphs between individuals. Our implementation follows the algorithm described in [13] closely, but we have generalized it to allow the exchange of subgraphs of unequal size. In Fig. 3 we present examples attained trough crossover.

We use a total of eight mutation operators: `Startshape mutate` – randomly selects a new V starting symbol; `Add V` – Adds a V symbol to a given production rule in a valid random position; `Remove V` – Removes a V symbol from a given production rule and associated parameters (if any exist); `Copy rule` – Duplicates a production rule; `Remove rule` – Removes a given production rule updating the remaining rules when necessary (if it is the only production rule associated with a given V symbol, production rules that generate that symbol must be updated, which is accomplished by removing the symbol from those rules); `Change`, `Remove`, `Add parameter` – as the names indicate these operators add, remove or change parameters, i.e. Σ symbols.

All operators preserve the validity of the grammar and update the graph accordingly.

4.3 System Overview and Generation Abilities

In all the experiments presented in this paper we adopted an user-guided evolution approach. Unlike most EA representation schemes, it is feasible to edit

CFDG by hand, in fact Context Free users have already created an impressive collection of shapes and visual languages. As such, it would be possible for the user to directly manipulate the genetic code. Although the ability to read, understand and edit the evolved genotypes is an important advantage, in the experiments presented herein we didn't take advantage of this possibility.

In standard EA systems the initial population is either random or seeded using examples from previous EA runs. In the present system, and given the availability of a wide set of hand-coded CFDG grammars, we have the option of using top-quality grammars to seed the evolutionary runs.

The remaining aspects of the system follow standard Evolutionary Computation practices. We use a generational approach, elitism and tournament selection. In the experiments presented in this paper, population size=50, the top individual was preserved, and tournament size=5.

In what concerns the generative potential of the system, it is trivial to show that it is possible to represente any image. Consider you want to represent a particular image, for each pixel use a rule that changes the color so that it matches the pixel's color, draw one square, move in the direction of the next pixel, call the rule for the next pixel. Obviously this would result in an extremely long and mostly useless grammar, but it can be done. Another way of demonstrating the generality of CFDG is the following: considering that an IFS can be specified (compactly) using CFDG and that Barnsley [12] demonstrated that IFSs can be used to generate any image, the same applies to CFDG. Although this generic representation abilities are theoretically relevant, in practice the main issue is knowing what types of images can be represented compactly. The wide set of imagery produced by Context Free users indicates that it is possible to generate a large amount of complex and beautiful shapes with surprisingly small grammars.

A more interesting question is knowing which languages of shapes can be expressed using CFDG. Once again theory and practice can be quite different.

From a theoretical standpoint, the set of all images of a given resolution, albeit vast, is finite [7]. As such, considering a fixed resolution, any given shape language is also a finite collection of shapes. Since it is possible to represent any image with CFDG and that union is a trivial operation for context free languages, it follows that any family of shapes can be represented[2].

In practice, defining a language of shapes through enumeration is either unfeasible or uninteresting. Thus, from a practical standpoint we can consider the set of all images and the set of all shape languages infinite. It then follows that it is not possible to represent any language of shapes, nor even the set of all recursive languages, since the pumping lemma for context free languages aplies. Like previously, the hand-coded examples of CFDG languages developed by the numerous users of Context Free indicate that, in spite of this limitations, it is possible to create interesting and sophisticated shape languages with compact CFDG grammars.

[2] Consider that you have two grammars, A and B, with initial symbols S_A and S_B, $A \cup B$ can be attained by: preserving all the production rules of A and B; creating a new initial symbol, $S_{A \cup B}$; adding the production rules $S_{A \cup B} \rightarrow S_A$ and $S_{A \cup B} \rightarrow S_B$.

5 Experimentation

The main goals of the experiments conducted were testing the ability of the system to: (i) evolve appealing and complex shapes; (ii) cope with hand-coded CFDGs; (iii) evolve families of shapes.

The analysis of the experimental results attained by evolutionary art systems, specially user driven ones, entails a high degree of subjectivity. In our case, there is an additional difficulty: each individual encodes a set of images. Considering these difficulties, space restrictions, and the visual nature of the results, we chose to focus on a single evolutionary, which can be considered typical.

To address goal (ii) we initiate the run using 6 hand-coded CFDGs downloaded from the Context Free gallery [11]. Fig. 4 presents examples of images created by these grammars.

In what concerns goals (i) and (iii) we already know it is possible to create stunning imagery and visual languages using CFDG, so the main issue is determining if it is possible to guide the EC algorithm to promising areas of the search space.

The visual diversity of the populations found throughout the run was always high (see Fig. 5). Population diversity is generally welcomed and has the additional benefit of keeping the user engaged; however, we found that the user was often distracted due to the presence of too many interesting alternatives, and unable to keep steady evaluation criteria. Nevertheless, the user was able to guide the EC algorithm with relative ease and promote convergence whenever it was found necessary. In Fig. 6 we present some of the favorite images produced by individuals of this run.

The mutation operators proved valid throughout the run producing results that are conceptually similar to the effects of mutation in expression-based EA. That is, the effects of mutation range from minor visual alterations to dramatic changes in appearance induced by small changes of the genetic code, with the later being less often [6]. Although it is subjective to say it, in what concerns mutation, the system appears to have an adequate degree of plasticity – allowing change – and stability – preventing chaotic behaviour.

The effects of the crossover operator appear to depend heavily on the structural similarity of the genotypes and on their size. In general terms, when the parents are unrelated the visual appearance of each descendent tends to be mostly determined by one of the parents (see fig. 3). This effect is particularly visible when the genotypes are small and with hand-built grammars (which tend to share little resemblance). Like previously, similar findings have been reported for expression-based EA, particularly for non-random initial populations [6].

In Fig. 7 we present instances of the visual languages defined by two of the individuals evolved during the run. The experimental results show that non-trivial and interesting families of shapes were evolved.

During evolution the user only has access to one instance of the images an individual generates. This means that the quality, diversity and consistency of the language of shapes generated by the individual isn't directly assessed. Arguably, individuals that fail to reliably generate high quality images will eventually be

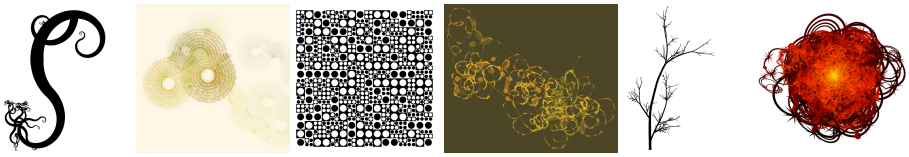

Fig. 4. Hand-coded grammars used as initial population.

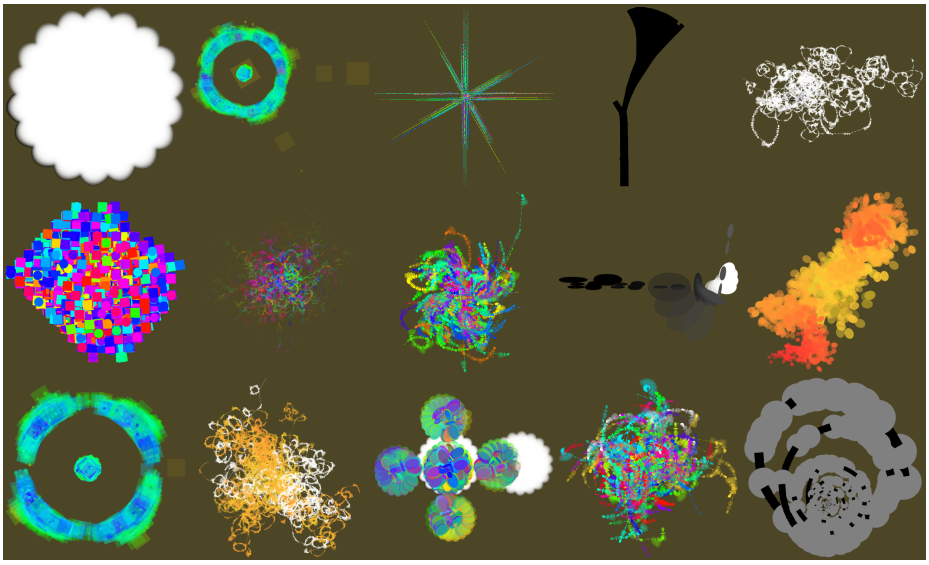

Fig. 5. Images generated by the first 15 individuals of the 10^{th} population of the run.

discarded by evolution, and the user will eventually grow tired of individuals that systematically generate the same image. Nevertheless, this is not the same as directly assessing the language of shapes that each individual defines, and interesting languages may easily be overlooked. This may contribute to a lower diversity of shapes within each family. In spite of this, the ability to create families of shapes is inherent to the the system and the experiments successfully evolved interesting visual languages.

6 Conclusions and Future Work

We presented a novel evolutionary engine that allows the evolution of CFDGs, is able to cope with non-deterministic grammars, and allows their recombination trough a graph-based crossover operator. Due to these abilities, it successfully overcomes the limitations of previous EC approaches where CFDGs are used. When compared with typical expression-based and parametric evolution models our approach presents several advantages, including the ability: to evolve visual

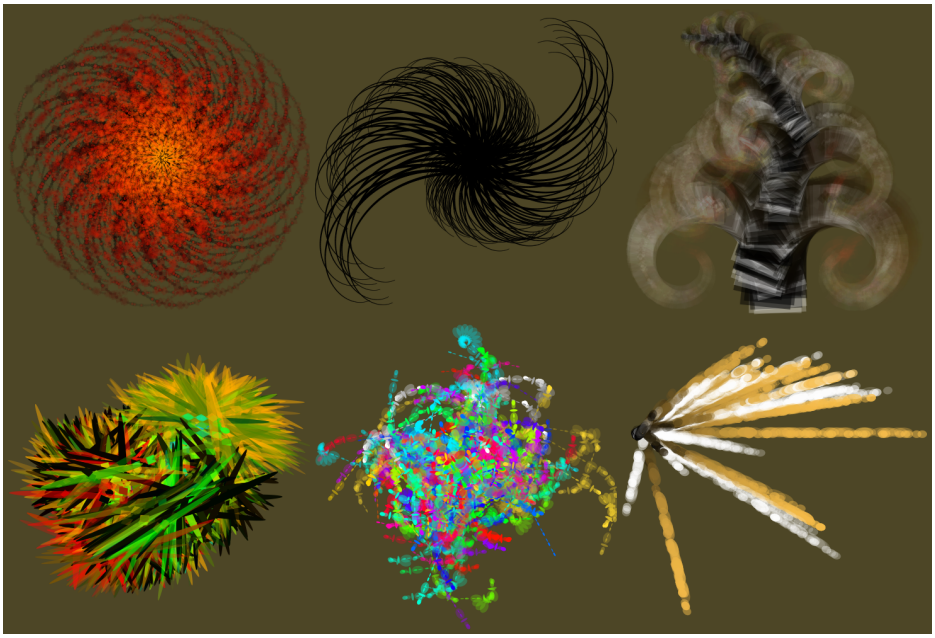

Fig. 6. Images generated by some of the most valued individuals evolved during the course of the run.

languages instead of individual images; to use hand-coded grammars; and to allow the user to editing of the genotypes.

Although the interpretation of the results is subjective, they provide evidence of the adequacy of the genetic operators and of the generative power and potential of the system. They also indicate that further experimentation is required to fully explore the potential of the approach for the creation of visual languages. Nevertheless, we consider this to be an important step in that direction.

In terms of future work, redesigning of the user interface, exploring of automatic image fitness assignment schemes, and developing approaches to automatically assess a language of shapes in terms of consistency, diversity and aesthetic qualities of the generated images are our top priorities.

References

1. Stiny, G., Gips, J.: Shape grammars and the generative specification of paintings and sculpture. In Freiman, C.V., ed.: Information Processing 71, Amsterdam, North Holland Publishing Co. (1971) 1460–1465
2. Boden, M.A.: The Creative Mind: Myths and Mechanisms. Basic Books, New York (1990)
3. Coyne, C.: Context free design grammar. http://www.chriscoyne.com/cfdg/ (last accessed in September 2009)
4. Lewis, M.: Evolutionary visual art and design. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Springer Berlin Heidelberg (2007) 3–37

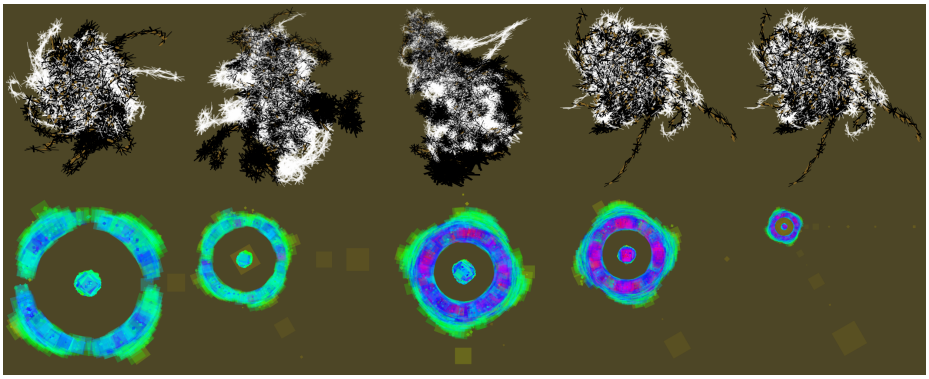

Fig. 7. Instances of the language of shapes defined by two individuals.

5. Sims, K.: Artificial evolution for computer graphics. ACM Computer Graphics **25** (1991) 319–328
6. Machado, P., Cardoso, A.: All the truth about NEvAr. Applied Intelligence, Special Issue on Creative Systems **16**(2) (2002) 101–119
7. McCormack, J.: Facing the future: Evolutionary possibilities for human-machine creativity. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Springer Berlin Heidelberg (2007) 417–451
8. Machado, P., Romero, J., Manaris, B.: Experiments in computational aesthetics: An iterative approach to stylistic change in evolutionary art. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Springer Berlin Heidelberg (2007) 381–415
9. Borrell, A.: Cfdg mutate. http://www.wickedbean.co.uk/cfdg/index.html (last accessed in September 2009)
10. Saunders, R., Grace, K.: Teaching evolutionary design systems by extending "context free". In: EvoWorkshops '09: Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing, Berlin, Heidelberg, Springer-Verlag (2009) 591–596
11. Horigan, J., Lentczner, M.: Context free. http://www.contextfreeart.org/ (last accessed in September 2009)
12. Barnsley, M.F.: Fractals Everywhere. Second edn. Academic Press Professional, Cambridge, MA (1993)
13. Pereira, F.B., Machado, P., Costa, E., Cardoso, A.: Graph based crossover — A case study with the busy beaver problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (13-17 July 1999) 1149–1155