

Teaching Evolutionary Design Systems by Extending “Context Free”

Rob Saunders and Kazjon Grace

Sydney University, Sydney NSW 2006, Australia,
rob@arch.usyd.edu.au, kazjon.grace@usyd.edu.au,
<http://web.arch.usyd.edu.au/~rob>

Abstract. This document reports on a case study using a novel approach to teaching generative design systems. The approach extends Context Free, a popular design grammar for producing 2D imagery, to support parametric and evolutionary design. We present some of the challenges that design students have typically faced when learning about generative systems. We describe our solution to providing students with a progressive learning experience from design grammars, through parametric design, to evolutionary design. We conclude with a discussion of the benefits of our approach and some directions for future developments.

1 Introduction

An understanding of the principles of *generative design systems* is an important component of any modern education in computer-aided design. The application of generative design technologies to solve complex real-world design problems has come to public attention with the development of the “bird’s nest” stadium for the Beijing Olympics [1]. Generative design systems are also increasingly finding uses in the “mass customisation” of goods and services, where they can be used to produce custom designs for anything from clothing to jewellery to housing. Technologies used to create generative design systems including a range of computational processes including evolutionary systems.

1.1 Teaching Generative Design Systems

As part of the Bachelor in Design Computing at the University of Sydney, we have been teaching generative design systems for more than five years, to both undergraduate and graduate students. These students typically have a strong design focus and, while their courses include programming, usually do not possess the breadth or depth of a computer science student. The curriculum for Generative Design Systems introduces concepts including; Expert Systems, Design Grammars, Parametric Design, and Evolutionary Design.

In the past we have had students implement their own evolutionary design tools based on a skeleton implementation. Based on student feedback, it was evident that the necessity to implement custom design domains significantly limited

student engagement. Students cited confusion caused by the number of “new” concepts introduced by evolutionary design systems. Many students were able to conceptualise and design the problems and fitnesses they wished to implement but lacked the technical ability to implement them. We also observed that using dedicated tools to teach individual generative design concepts created a barrier to students integrating concepts towards a more complete understanding.

To address these issues we have used and extended the Context Free design tool to provide a single environment to learn about three important concepts; design grammars, parametric design, and evolutionary design. The remainder of this paper presents how we have used this system provide a path for learning about design grammars, parametric design and evolutionary design through a series of simple extensions of the initial grammar.

2 Description

The Context Free Design Grammar (CFDG) is a simple language designed to generate complex images from simple programs [2]. Context Free is an open-source graphical development environment for writing and rendering CFDG programs [3]. Informally, a CFDG program contains a set of simple rules describing how to draw designs using a small number of pre-defined terminal shape rules including `circle`, `square`, and `triangle`. Recursive grammars include rules that call themselves. Non-deterministic grammars include multiple rules with the same name, the rule to execute is chosen probabilistically. An example of a recursive, non-deterministic CFDG program together with one of the images it can produce is given in Fig. 1.

Other graphical grammar-based design tools such as LParser [4] or Xfrog[5] have more powerful rendering engines and can produce 3D models as well as 2D images. The advantage of Context Free is its simple syntax and intuitive user interface. Our experience has been that students are able to explore the generative potential of recursive and non-deterministic rules and rapidly develop proficiency by modifying, extending and creating Context Free grammars despite their limited programming experience.

2.1 Parametric Context Free

By definition, CFDG does not support variables or parameters; the inclusion of variables would violate the need to keep the CFDG rules free of context. To allow students to explore parametric design grammars we first created an interpreter for parametric design grammars that uses template files to define a set of parameters and a set of modified CFDG rules.

An example template is shown in Fig. 2a. This template defines ten parameters and three rules: the rule `LINES1` calls `LINES2` and then recursively calls itself, similarly `LINES2` calls `LINE` and then recursively calls itself, `LINE` draws a short line segment using the primitive `TRIANGLE` rule that draws an equilateral triangle. Parameters are defined within the comments tags at the top of the file

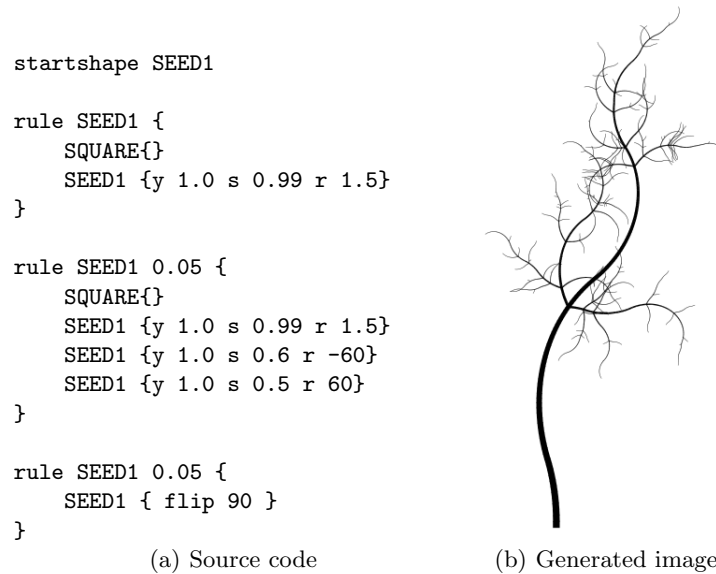


Fig. 1. An example Context Free Design Grammar and one of the images that it can generate. Context Free uses short letter strings to represent its random number seeds. This image was generated using the string MKY.

and referenced throughout the rules to define scales, rotations, and translations. Standard CFDG grammar files are generated by substituting all references to a parameter with a value within the range defined for the parameter. Some of the possible outputs for the parametric CFDG in Fig. 2a are shown in Fig. 2b. The space of possible designs generated from the three CFDG rules (comprising just seven lines of code) is very large, illustrating how quickly students could generate interesting design spaces to explore with the parametric CFDG system.

2.2 Evolutionary Context Free

To allow students to evolve design grammars, an evolutionary CFDG system was developed using the parametric CFDG system. In the evolutionary CFDG system, a parameter set in a template file defines the genotype for individuals and the rules are used to express genotypes into phenotypes. An interactive evolutionary system was first developed, where the phenotype (image) for each individual in a population is displayed side-by-side. The interactive evolutionary CFDG system displays each population of image-based phenotypes to allow users to select one or more parent designs by clicking on them with the mouse. The evolutionary CFDG system uses a standard one-point crossover operator and per-gene mutation to generate children from parents.

```


/* $RA = [-1,1] */
/* $XA = [-1,1] */
/* $YA = [-1,1] */
/* $RB = [-360,360] */
/* $SB = [0.85,0.99] */
/* $RC = [-1,1] */
/* $XC = [-1,1] */
/* $YC = [-1,1] */
/* $RD = [-360,360] */
/* $SD = [0.85,0.99] */

startshape LINES2
rule LINES2 {
  LINES1 { r $RA x $XA y $YA }
  LINES2 { r $RB s $SB }
}

rule LINES1 {
  LINE { r $RC x $XC y $YC }
  LINES1 { r $RD s $SD }
}

rule LINE { TRIANGLE { s 0.025 1 }}

```



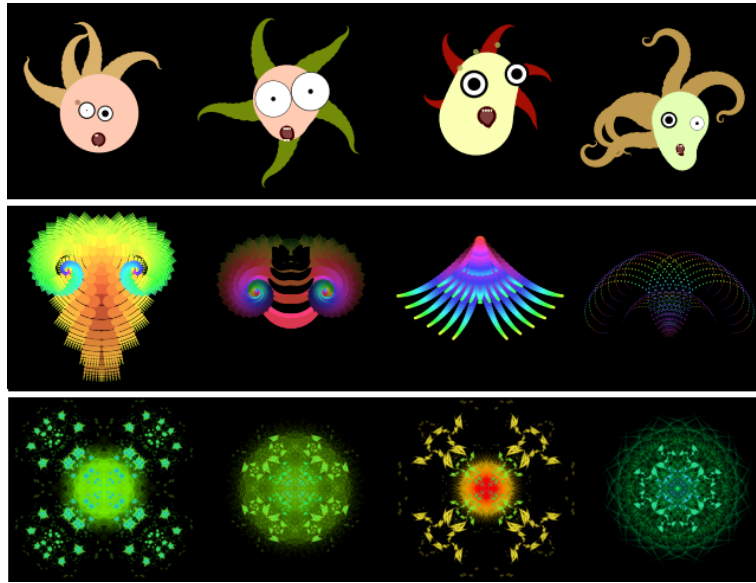
(b) Renders

Fig. 2. A parameterised CFDG template with three rules and ten variables.

The evolutionary CFDG system was developed in Processing [6], a programming environment that the students were already familiar with. The code to the application was made available to the students allowing them to experiment with writing fitness functions to replace user selection. Some fitness functions such as surface area, X/Y symmetry, rotational symmetry, colour variance and brightness were provided as examples. Students created new fitness functions to guide the evolution of their parametric CFDGs.

3 Examples of student work

Examples of the outputs from evolutionary design grammars developed by students can be seen in Fig. 3a, giving some indication the breadth of possible design spaces. These examples were generated using the interactive evolutionary CFDG system. An example evolution of a design grammar using an image-based fitness function written by a student is given in Fig. 3b. The fitness function counts the number of colour changes by scanning horizontally across the centre of the image. This fitness function promotes complexity and in this example results in a design using tightly-packed, spiralling squares with alternating colours.



(a) Designs evolved using three student-developed parametric grammars.



(b) Designs evolved based on a student-developed fitness function.

Fig. 3. Designs from (a) interactive and (b) non-interactive evolutionary systems.

Using the extensions to Context Free, students were introduced to design grammars, parametric grammars, interactive evolutionary grammars, and finally non-interactive evolutionary grammars. This progression gave the students the opportunity to first learn about constructing design spaces before learning methods for searching them. All students demonstrated some understanding of how the definition of parameters as genotypes and rules as the means of expression worked together to define a design space.

Students were able to communicate the concepts they wanted to evolve using fitness functions, e.g., spikes, appendages, roundedness, gradients, pleasant colour combinations, produced documentation showing they possessed a good understanding of how GAs work and can be applied. We observed a varying degrees of success when they came to develop those concepts into algorithms. A minority of students were able to implement their fitness functions. Other students had difficulty following the implementations of the provided fitness functions but were able to apply and combine them to achieve results that they found interesting.

4 Discussion

The interactive evolutionary CFDG system proved an effective way to teach design students the principles of design grammars, parametric design and evolutionary design. In contrast with previous teaching approaches, which involved students experimenting with configurable but fixed problems for each class of design system, the use of CFDG templates allowed students to implement their own problem domains very quickly.

The parametric template system, in which parameters can be inserted into any design grammar and interpreted as genes, allowed students to experiment with altering design spaces and immediately see how they affect the search process. This allowed students to experiment with the definition of their own genotypes, their own genotype-to-phenotype expressions and their own fitness functions. This allowed students to gain a deeper understanding of how the design of problems, representations and fitness functions affects the evolutionary design process. Students were able to experiment with the development of evolutionary systems with relatively little programming experience. Compared to other methods of teaching evolutionary design systems we found this method to be highly effective and enjoyable for the students to learn and experiment with.

Based on feedback from students, we believe our approach, based on a series of extensions to Context Free, shows great potential in the teaching of generative design systems to design students. Future iterations of this teaching method may benefit from an increased focus on the graphical user interface to the extended system. For example, to ease the implementation of fitness functions, students may benefit from the addition of a graphical interface to select and combine a set of existing fitness functions, as an alternative to writing code. This would further lower the exposure of the students to the underlying implementation while opening up the possibility for exploring the consequences of fitness function design to more students.

References

- [1] Glancey, J.: Secrets of the Birds' Nest, The Guardian Online. <http://www.guardian.co.uk/artanddesign/2008/feb/11/architecture.chinaarts2008>, last accessed January 2009.
- [2] Coyne, C.: Context Free Design Grammar. <http://www.chriscoyne.com/cfdg/>, last accessed January 2009.
- [3] Horigan, J., Lentczner, M.: Context Free. <http://www.contextfreeart.org/>, last accessed January 2009.
- [4] Lapré, L.: Lparser. <http://members.ziggo.nl/laurens.lapre/lparser.html>, last accessed January 2009.
- [5] Lintermann, B., Deussen, O.: Xfrog. <http://www.xfrog.com/>, last accessed January 2009.
- [6] Reas, C., Fry, B.: Processing: A Programming Handbook for Visual Designers and Artists. The MIT Press, 2007.