Evolutionary design using grammatical evolution and shape grammars: designing a shelter

Michael O'Neill*, James McDermott, John Mark Swafford, Jonathan Byrne, Erik Hemberg, Anthony Brabazon

Natural Computing Research & Applications Group
Complex & Adaptive Systems Lab
University College Dublin, Ireland
E-mail: m.oneill@ucd.ie
E-mail: jamesmichaelmcdermott@gmail.com
E-mail: johnmarksuave@gmail.com
E-mail: jonathanbyrn@gmail.com
E-mail: erik.hemberg@ucd.ie
E-mail: anthony.brabazon@ucd.ie
*Corresponding author

Elizabeth Shotton, Ciaran McNally

School of Architecture, Landscape & Civil Engineering
University College Dublin,Ireland
E-mail: elizabeth.shotton@ucd.ie
E-mail: ciaran.mcnally@ucd.ie

Martin Hemberg

Department of Ophthalmology, Children's Hospital Boston, 300
Longwood Avenue, MA 02215, USA
E-mail: martin.hemberg@childrens.harvard.edu

Abstract: A new evolutionary design tool is presented, which uses shape grammars and a grammar-based form of evolutionary computation, grammatical evolution (GE). Shape grammars allow the user to specify possible forms, and GE allows forms to be iteratively selected,

recombined and mutated: this is shown to be a powerful combination of techniques. The potential of GE and Shape Grammars for Evolutionary Design is examined by attempting to design a single-person shelter to be evaluated by collaborators from the University College Dublin school of Architecture, Landscape, and Engineering. The team was able to successfully generate conceptual shelter designs based on scrutiny from the collaborators. A number of avenues for future work are highlighted arising from the case study.

Keywords: Evolutionary Design, Architecture, Grammatical Evolution, Shape Grammars, Genetic Programming, Evolutionary Computation

Biographical notes: Michael O'Neill is a Senior Lecturer with the School of Computer Science & Informatics, and the Complex & Adaptive Systems Lab at University College Dublin. He is is co-founder and co-director of the UCD Natural Computing Research & Applications Group, and has authored a number of books including 'Grammatical Evolution', 'Foundations in Grammatical Evolution for Dynamic Environments' and 'Biologically Inspired Algorithms for Financial Modelling'.

James McDermott is a post-doctoral researcher with UCD's Natural Computing Research & Applications Group. His PhD thesis was on Interactive Evolutionary Computation applied to Sound Synthesis. His research interests include representations for aesthetic EC and EC applications in music and design.

John Mark Swafford, Jonathan Byrne and Erik Hemberg are PhD students with UCD's Natural Computing Research & Applications Group.

Elizabeth Shotton is a Lecturer in UCD School of Architecture, Landscape & Civil Engineering, and also maintains a professional architectural practice in Canada and Ireland.

Ciaran McNally is a Senior Research Engineer in the Bridge & Transport Infrastructure Research Group at UCD. He is Principal Investigator on a series of EU funded research projects into infrastructure management and coordinator of the international Marie Curie Initial Training Network TEAM - Training in European Asset Management.

Anthony Brabazon is currently a Professor and Head of Research in the School of Business at University College Dublin. He is also co-founder and co-director of the UCD Natural Computing Research & Applications Group. His primary research interests concern the development of Natural Computing theory and the application of Natural Computing in Finance and beyond.

Martin Hemberg is a post-doctoral researcher at the Department of Ophthalmology at Children's Hospital Boston. He obtained is PhD from Imperial College London and he has also worked at the Architectural Association in London. His primary research interests include mathe-

matical and computational models of gene expression

---

1   Introduction

The natural process of biological evolution has clearly demonstrated its power to design elegant form and structure in the name of survival. As such, it is natural to turn to algorithms which are inspired by this process to tackle design problems. These evolutionary algorithms (EAs) are powerful problem-solving tools. Of particular note are the genetic programming (GP) variants which are now capable of routine human-competitive performance, e.g. in the area of analog circuit design (Koza, 2003). Some evolved solutions have passed human tests of innovation by being patentable in their own right. The long term objective of the research, of which this study forms the seed, is to explore and extend a powerful and already well recognised grammatical GP approach—grammatical evolution (GE)—to challenging architectural design problem environments. This will be achieved through the adoption of a rich and evolvable representation, shape grammars (Stiny, 1980). Shape grammars allow a natural way to encode human domain knowledge into the evolutionary/generative process, and are proving invaluable in the area of generative design. An example is the Integrated Design Innovation Group at Carnegie Mellon University who explore the essence of the design of products ranging from Harley Davidson motorcycles to cars and coffee makers (Cagan and Vogel, 2001; Vogel and Cagan, 2005). To date, the shape grammar formalism has not been combined with an advanced grammatical evolutionary algorithm such as GE, and the current research addresses this important gap.

The remainder of this paper is structured as follows. The following Section 2 provides some general background and motivation. Next is an introduction to interactive evolutionary design, followed by an overview of representations adopted in the evolutionary computation (EC) literature for design, including a description of the shape grammar formalism. A brief introduction to GE is given in Section 4. Section 5 contains a description of a specific design problem and a specific shape grammar suitable for it, and a description of results obtained. Finally, Section 6 gives conclusions and future work.

2   Background

EC, and in particular GP, have clearly demonstrated their potential in the broadly-defined real world application domain of design. They have produced solutions that are competitive with, and in some cases superior to, those developed by human experts, and which have resulted in patentable inventions (Koza, 2003; Takagi, 2001; Bentley, 2000; O'Neill and Brabazon, 2009). As such, this domain (in particular analog circuit design (Koza, 2003)) has been a proving ground for the capabilities of an artificial evolutionary process, and has led to arguably the first routinely human-competitive form of machine learning. The combination of

an evolutionary algorithm and a grammatical representation (or design language) is a particularly powerful and novel departure of recent years (Hornby, 2005). Examples of research in this intersection of fields include the Genr8 and GENRE systems (Hornby, 2005; Hemberg and O'Reilly, 2004; O'Reilly and Hemberg, 2007; Gero et al., 1994). The grammar-based form of GP as realised in GE has a number of advantages over more traditional optimisation methods for the design domain (Hornby, 2005; Hemberg and O'Reilly, 2004; O'Neill and Brabazon, 2008). Some of these advantages are as follows.

- GP handles the search of open-ended structure. The model size and structure are not specified a priori.

- Search is stochastic. The exploratory process is not limited or biased by the imagination of the human user or preconceptions based on prior typologies or known solutions. Therefore designs can be produced which are novel and sometimes counterintuitive, which has significant implications in the architectural design process. Examples include recent work by architectural practices such as Greg Lynn and Foreign Office Architects (2009).

- The unbiased nature of stochastic evolutionary search can be balanced by introducing a desired bias in the form of architectural domain knowledge. This can be achieved by allowing the user to specify proven structural forms, planning constraints, and even aesthetic preferences a priori. In a grammar-based form of GP such as GE, domain knowledge can be easily incorporated through the underlying grammatical representation.

2.1   Interactive Evolutionary Computation

Human interaction was originally introduced to the evolutionary process for problems where no suitable fitness function could be found. Interactivity opened up a new domain for evolutionary computation, problems that required aesthetic judgment. This domain has since become one of the main motivations for IEC, as aesthetic problems have a subjective element in their evaluation that is difficult to define in an algorithmic fitness function. Human interaction has allowed EC to be applied to problems such as music and computer graphics, and to act as an exploratory tool as opposed to merely being an optimiser. One of the earliest attempts to introduce human evaluation was the work by Richard Dawkins called Biomorphs (Dawkins, 1986). This work simulated the evolution of 2-D branching structures made from sets of genetic parameters, where the user selects the individuals for reproduction.

After Biomorphs there was an increase of research in the field of both interactive genetic algorithms (IGA) and interactive genetic programming (IGP). The seminal paper by Karl Sims (Sims, 1991) showed that basic user interaction was capable of creating complex and beautiful artwork. Sims used human interaction to create images, three-dimensional textures, and, by adding an extra dimension for time, animation. IGAs have since been applied to fields as diverse as music generation (Biles, 1994; McDermott, 2008), anthropomorphic symbols (Dorris et al., 2004), 3-D lighting (Aoki and Takagi, 1996), and aircraft frames (Parmee and Bonham, 2000). Interactivity has also been used in conjunction with GP for form

design. This has led to the development of several form design tools (O'Reilly and Ramachandran, 1998; Bentley and O'Reilly, 2001). A more complete list of applications of interactivity can be found in the literature by Banzhaf (1997) and Takagi (2001).

3   Grammatical Representations In Evolutionary Design

In this section, previous approaches to grammatical representations in evolutionary design are discussed. This overview will cover shape grammars and how they have been adapted to aid in the evolutionary design process as well as other grammatical approaches to solving design problems.
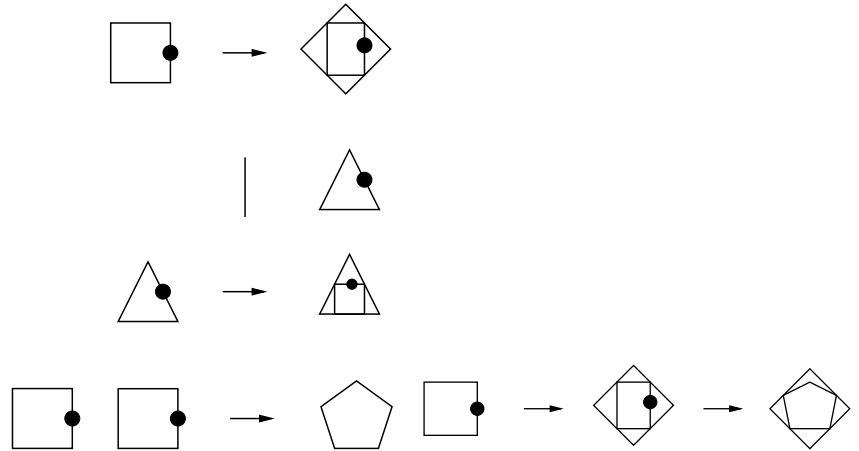
3.1   Shape Grammars

Shape grammars were first formally defined by Stiny (1980), where a shape is defined as: a limited set of lines called "maximal lines" that may be plotted on a two or three dimensional graph. These shapes may be subshapes of or identical to other shapes. The boolean operations of union, intersection, difference, and complement may be applied, as well as Euclidean transformations such as translations, rotations, reflections, scalings, and compositions. Stiny's shape grammars are intended to be a symbolic analogue to familiar string-rewriting grammars. They are generative grammatical re-writing systems with terminal symbols, non-terminal symbols, a start symbol, and production rules expressing how non-terminals may be replaced with other terminals and non-terminals. A simplified shape grammar definition is as follows:

1. S, a finite set of shapes

2. L, a finite set of symbols or labels

3. R, a finite set of shape rules having the form $a \rightarrow \beta$, where a is a labelled shape in the set $(S, L)^+$, and β is in the set $(S, L)^*$

4. I the initial, nonempty labelled shape.

A graphical example of a shape grammar and a sample derivation are given in Figures 1(a), 1(b) and 1(c).

3.2   Work Inspired By Shape Grammars

Much work has been inspired by this first definition of a shape grammar, as researchers have found practical applications in the areas of art, engineering, and architecture. For now, the focus of this work will be on applying shape grammars to engineering, planning, and architecture. One of the first applications of shape grammars in architecture was in work concerning Frank Lloyd Wright's prairie houses (Koning and Eizenberg, 1981). Here, shape grammars were used to generate new and different house plans in Wright's style. Another, more recent use of shape grammars in architecture can be seen in the work of Wonka et al. (2003) towards

(a) Shape Grammar Initial Shape

(b) Example Shape Grammar. The 'right arrow' shows what the left-hand nonterminal may be replaced with. The vertical bars are separators for the different productions within a single rule.

(c) Example Shape Grammar derivation sequence and resulting output

rapid architecture generation. Set grammars (shape grammars featuring higher-level primitives) and split grammars (shape grammars constraining the developed shape inside a fixed space to avoid overlap) were employed to quickly generate valid designs. More work that has been inspired by Stiny's shape grammars are landscape grammars described in the work of Mayall and Hall (2005, 2007). These landscape grammars are capable of generating landscapes based on landscape object types (houses, trees, fences, etc.) and spatial syntax rules (commercial areas, forested areas, paths for roads, relationships between these areas, etc.). The surface generation tool, GENR8 (Hemberg and O'Reilly, 2004; Hemberg et al., 2007b), is another popular example of work in grammatical representations and evolutionary design. This tool uses GE combined with modified L-Systems, Hemberg Extended Map L-Systems (HEMLS), to generate and evolve surfaces. It is also possible to modify or evolve a shape grammar during an evolutionary run. One example of this used a shape grammar and a genetic algorithm to design a section of a beam (Gero et al., 1994). But rather than only evolving the binary arrays that typically represent individuals in a genetic algorithm, they evolved the shape grammar as well. This led the shape grammar to modify the search space and evolution was able to explore more design possibilities. The work presented here examines the effects of different grammars and grammatical representations on the three dimensional design representations that have also been inspired by Stiny's definition of a shape grammar (Stiny, 1980). The grammars used here are also simplified representations due to the complexity of implementing a pure shape grammar. More detail on this will be shown in Section 5. It is also important to recognize the prior attempts at using evolutionary computation to optimize design problems that did not involve grammatical representations. Many of these approaches use genetic algorithms while fewer use GP to approach their design challenges. The reader may find much

information on this topic in the survey by Kicinger et al. (2005). We now continue with a brief introduction to the GE algorithm and how shape grammars can be adopted with this grammar-based form of GP.

4   Grammatical Evolution

Grammatial Evolution (GE) (Dempsey et al., 2009; O'Neill and Ryan, 2003; O'Neill, 2001; O'Neill and Ryan, 2001; O'Neill et al., 2003; Ryan et al., 1998) is a grammatical approach to GP, and there is a vast literature on the use of grammars in GP— see, e.g. (Hicklin, 1986; Poli et al., 2008; Whigham, 1996; Wong and Leung, 2000; Ratle and Sebag, 2000; McKay et al., 2005). GE has been used in a range of application domains, from financial modeling to music (Brabazon and O'Neill, 2006; Dempsey, 2007; Reddin et al., 2009)[a].

```
<scene> ::= <shape> | <scene> + <shape>
                                                (2 productions)
<shape> ::= wall | slab | <path> | connect_point(<path>)
        | connect(<path>, <path>) | connect_parallel(<path>)
        | perpendiculars(<path>)
                                                (7 productions)
<path>  ::= line | circle | bezier | spiral(<path>)
                                                (4 productions)
```

Figure 1   An example grammar.

In GE each individual is internally represented as an array of integers, a genome. GE uses a generative grammar to guide the construction of a phenotype string output from this genome input. We briefly illustrate an example of this mapping using the example grammar outlined in Figure 1. Although the grammar refers to shapes, the mapping is purely formal (syntactic) string-rewriting. No understanding of the semantics of the strings or of shape grammars is required. We begin the development of an individual from a seed known as the start symbol, typically the first non-terminal symbol in the grammar file. In this case starting from `<scene>` there are two possible productions, which can transform it into either `<shape>` or into `<scene> + <shape>`. To decide which production replaces `<scene>`, we consult the input genome: that is, we read the next available (integer-valued) codon and apply the following mapping function:

$$production \; = \; c \, \% \, n$$

where `c` is the codon, and `n` is the number of productions (on the right-hand side) available for the current non-terminal (on the left-hand side). In the rule with `<scene>` on the left-hand side, there are two options, so given a codon value of 6, we choose production 6 % 2 = 0. In other words, `<scene>` will be replaced with `<shape>`. The mapping continues by taking at each step the left-most non-terminal symbol in the developing solution, examining the grammar to determine whether

---

[a]Further information on GE and pointers to code can be found at `http://www.grammatical-evolution.org` and `http://ncra.ucd.ie`

```
-------------------
| 6 | 19 | 13 | ...
-------------------


<scene> -> <shape>                     (6 % 2 = 0)
        -> connect_parallel(<path>)    (19 % 7 = 5)
        -> connect_parallel(circle)    (13 % 4 = 1)
```

Figure 2   Part of a sample GE individual's genome, and the mapping from genome to phenotype string. Each equation gives the codon value, the number of possible productions, and the index of the chosen production.

a choice has to be made for that non-terminal and, if so, reading the next codon. Continuing to read codon values from the example individual's genome in Figure 2 will give the expansion in Figure 2.

The above process is responsible for developing a single genome (an array of integers) into it's corresponding phenotype (a sentence in the language specified by the grammar). Thereafter, the GE algorithm is similar to a standard genetic algorithm or GA. The string is evaluated according to a fitness function to give a fitness value (some measure of the solutions quality). The fitness function may calculate a simple structural property, such as the similarity of the phenotype string to a target, or a complex property such as the expected return on investment of a financial trading rule derived from the phenotype string. In the case of design, it is possible to imagine fitness functions based on properties such as symmetry and weight (Hemberg and O'Reilly, 2004; Hemberg et al., 2007b), and it is also possible to have a user assign fitness values, as discussed in Section 2.1.

At the beginning of the GE algorithm, a population of individuals' genotypes is randomly generated. The genotype-phenotype mapping and calculation of fitness are repeated for every individual in the population. The individuals with the best fitness values are then selected, and their genomes are recombined and mutated, yielding a new population. Over many iterations of this process, the population evolves towards individuals with better fitness values, and ideally converges on a global optimum. GE extends the EC metaphor with natural evolution, by borrowing additional principles from molecular biology. The most powerful of these is the genotype-phenotype map explained above. Earlier research in GP has shown some of the potential benefits of such a mapping (Banzhaf, 1994; Keller and Banzhaf, 1996) and GE further exploits it to create a highly modular and flexible approach to program/model induction. An example of this is the fact that the model space can be explored by search engines other than the typical GA, such as particle swarm optimization and differential evolution (O'Neill and Brabazon, 2006a,b).

The flexibility of GE is such that even with the presence of the genotype-phenotype map, traditional tree-based search operators such as crossover and mutation can be adopted in place of the genotype search operators, effectively transforming GE into a standard form of GP with the grammar used during the initialization of the population (Harper and Blair, 2005). It is also possible to use both genotype- and phenotype-focussed search operators, combining the benefits of each approach. The next section describes a design problem to which GE was applied.

5 Shelter Design Case Study

An interactive, 3-D version of the grammatical evolution/shape grammar system was implemented as a plugin for the Blender 3D software (Stichting Blender Foundation, 2009), and tested by several of the authors on a specific design assignment.

5.1 Design Brief

The test problem was chosen as being sufficiently complex to study the GE design tool's capabilities in an initial trial, but not overwhelmingly complex in materials or structures. It was derived from a design brief assigned in 2009 to teams of students in the first year of the graduate program in the UCD Architecture and Civil Engineering programmes. Students were required to design and build a single-person shelter, i.e. a structure useful to give shelter from the weather to one or more people outdoors. Structures were to be composed solely of 1"x3" or 1"x2" ash beams, with the possibility of adding other sheet materials and fasteners where necessary. The original brief required the construction of full-scale prototypes, and so demanded attention to site, gravity, jointing details, and the construction process.

This assignment was re-interpreted, simplified, and scope-limited to be suitable as a test for the evolutionary software, and in light of the (current) users' lack of architectural training. The criteria for evaluation of the results were similar to those used to assess the work undertaken by the engineering and architecture students, though limited in this initial trial to issues of form and gravity only. Funes and Pollack (1997) have successfully demonstrated that it is possible to incorporate gravity into an Evolutionary Design process.

The task of actually building the final designs and making them site-specific were not considered. The question of overall scale was not dealt with, and designs shown here can be freely re-scaled as necessary. The joints between beams were not regarded as part of the task, and were not represented explicitly. Finally, the addition of sheet materials, if any, was regarded as a manual, post-evolutionary step. Thus the software was required only to represent and produce frame-like structures of thin beams, as described next.

5.2 Grammatical Representation

The simplest possible representation might be a recursively-grown list of independent beams, with their end-points each represented as (x, y, z) co-ordinates. A grammar of this type is given in Figure 3. Early experiments using such a representation confirmed that it is far *too* direct and fails to capture the kind of compositional structure appropriate to the domain: the overwhelming majority of designs consisted of multiple, unconnected beams, many of which appeared to be floating in mid-air (as shown in Figure 3). This representation was *sufficient* to produce any given design, but coherent, well-formed designs appeared infrequently in the search space, and evolution consisted largely of vain search for these properties rather than a gradual imposition of more interesting aesthetic properties. In

```
<scene> ::= <beam> | <scene> + <beam>
<beam>  ::= beam(<x>, <x>, <x>)
<x>     ::= 0 | 1 | 2 | ... | 9
```


Figure 3   A grammar producing multiple independently-placed beams, and a typical individual produced using this grammar.

IEC, in particular, it is necessary to express known constraints via the representation rather than the fitness function—to prevent ill-formed individuals from arising, rather than waste the user's time asking for evaluations of them.

A better representation will attempt to ensure that each individual's constituent beams are arranged coherently with respect to each other. A new grammar was developed based on two fundamental concepts: curves, and lists of beams created by operating on lists of points on these curves. In particular the grammar was intended to produce pleasant curves despite working only with straight beams.

A simplified version of the shape grammar is given in Figure 4, and a sample derivation using this grammar in Figure 5. Note that the grammar and derivation correspond exactly to the string-form grammar and derivation given in Figures 1 and 2. Note that the individual depicted in Figure 3 is still producible using this grammar. Additional examples of individuals that can be generated from this grammar are given in Figure 6 along with their corresponding phenotype strings.

5.3   Implementation and Experimental Setup

The software was implemented as a plugin for the Blender 3D software, and written in the Python programming language. Since IEC is seen as one tool in a designer's toolbox rather than a replacement for other tools (Bentley and O'Reilly, 2001), it is better to implement it as a plugin to an existing 3D system rather than a standalone program. The plugin provides a user-interface within Blender, and uses the existing GEVA software (O'Neill et al., 2008), written in-house by the NCRA.

The experimental parameters are given in Table 1. In interactive EC, the user is free to quit at any time, however, for the purpose of user evaluation an upper bound of 50 generations is imposed here. The population size, 15, is also an upper bound, since not every genome gives rise, after the GE mapping process explained in Section 4, to a valid phenotype. The other settings are quite typical of standard GE.
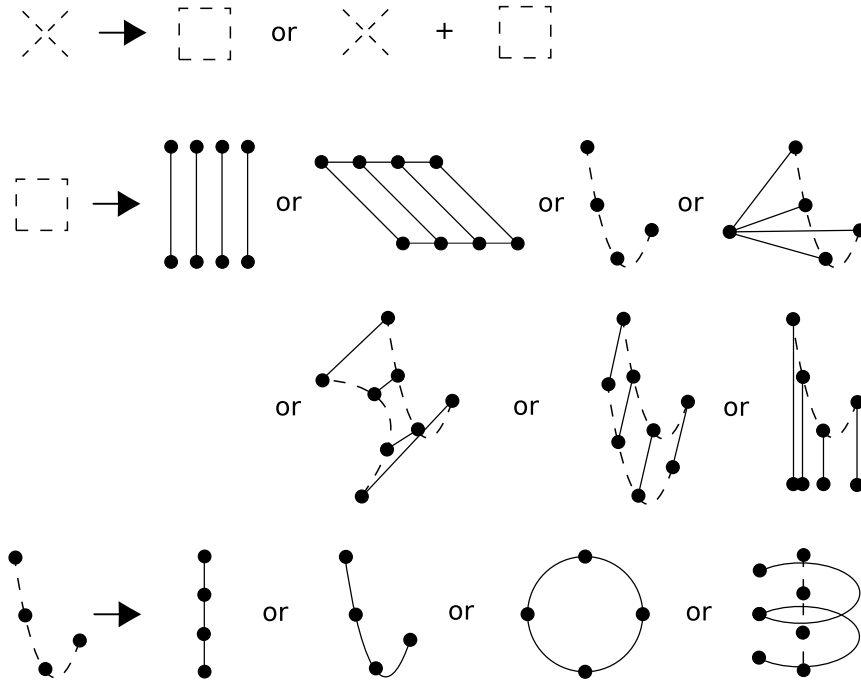
Figure 4   Shape grammar. The dashed X indicates the special non-terminal, the start shape. The dashed square is a non-terminal representing a single shape. Other dashed lines indicate non-terminal paths, which may be transformed as in the third rule. Terminals, in solid lines, can not be transformed by these rules, but are parameterised: size, orientation, start and end points, and so on, are determined by numerical parameters.
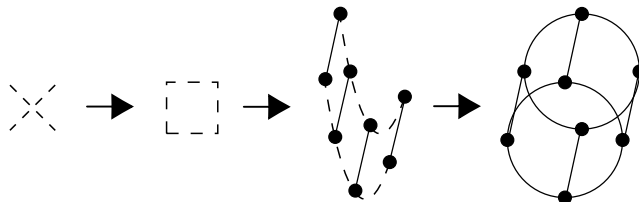
Figure 5   Sample derivation. Here, the start shape is transformed to the single-shape non-terminal; this is transformed to the sixth of the high-level primitives, an uninstanti-ated curve connected by beams to a displaced, parallel copy of itself. Finally, the curve is instantiated as a circle. As shown, the copy takes account of this instantiation.

5.4   Results

IEC in aesthetic domains does not lend itself to a typical analysis of success through numerical results and statistics. Instead, the software was used for multiple runs by multiple users attempting the shelter design task; results were evaluated subjectively, as set out in this section, by collaborators on the project from UCD Architecture, Landscape, and Civil Engineering. Also, specific changes to the configuration and representation have been made and tested.

(a) perpendiculars(bezier) + perpendiculars(bezier)

(b) connect_point(bezier)

(c) circle

(d) connect(spiral(line), spiral(line))

(e) wall + slab

(f) line + connect(bezier, bezier)

(g) connect(bezier, bezier)

(h) connect(bezier, bezier)

(i) connect(circle, line)

Figure 6   Sample individuals illustrating the primitives available in the grammar. Note that these were produced by hand-writing the phenotype strings given above, not through evolution.

Table 1   Parameter settings for the interactive 3D software.

Parameter	Value
integer codon mutation	0.02
initialisation	ramped-half-and-half
max generations	50
max population size)	15
selection	tournament
tournament size	3
replacement	generational
elitism	1 individual
1-pt crossover	0.7
max wraps	3
fitness	binary, interactively assigned

User 1 found, in early runs, that individuals featuring the *circle* primitive were not to his taste. This user therefore chose to eliminate this primitive from the grammar for later runs. The grammar is human-editable, and this is a viable mode of interaction, but it is optional: an understanding of the grammar is not required for normal use. The outcome of this change was that no structures featuring circles were produced, and so at each generation a higher number of potentially desirable structures were available. It would be desirable to make this interaction more of a "black box" so a user who is not familiar with grammars in any sense may make these types of modifications. Two final designs from user 1 are shown in Figure 7. The aim here was to produce shelters with walls on two sides to protect against varying wind and rain. The individual on the left was found in the first generation of the first trial. The multiple walls and two distinct roof shapes were considered to be practical and appealing. The individual on the right appeared in a different run, and again gives the desired enclosure effect.

Feedback from the collaborators included the suggestion that the multiple roofs in Figure 7(a), being cantilevered from one support beam and relatively heavy-looking, might require steel reinforcement. The individual in Figure 7(b) had problems with a very heavy and bulky roof. Also the enclosure would potentially be a wind-trap and lead to uplift in the roof. However, several features of the design were regarded as positive. It could be oriented to avoid the prevailing wind. Allowing rotation of one wall would make the design more flexible and useful.

(a)   (b)

Figure 7   Two designs from user 1.

In early runs, **user 2** found that too often an individual rated as bad in one generation would re-appear or give rise to offspring in the next. The hypothesis was formed that this problem was due to the combination of a binary interactive

fitness function and tournament selection. Tournament selection, typical in GE, works by randomly selecting n individuals ($n = 3$ in our experiments) from the population, adding the one with the best fitness to the mating pool for the next generation, and repeating until the mating pool is full. With binary fitness it is possible for a tournament to contain only bad individuals, and in this case it is possible for a bad individual to be added to the mating pool. The solution is to use roulette-wheel selection instead, in which the mating pool is created by randomly sampling the population weighted by fitness. User 2 made this change to the configuration, and found that it allowed a far more focussed type of exploration. A population consisting only of variations on a single individual could be produced at each generation.

User 2's finished design is shown from two angles in Figure 8. The spiral/spoke motif is aesthetically appealling, and the design's two sides give protection even against Ireland's horizontal rain. The roof allows a large amount of light through. The design was evaluated as quite successful. The roof was thought to be very heavy, and thus difficult to keep upright. The roof span is also quite large which could make it difficult to support, but its symmetry was thought to provide a possible solution: cables between opposing points in the roof might keep the roof intact and balanced.
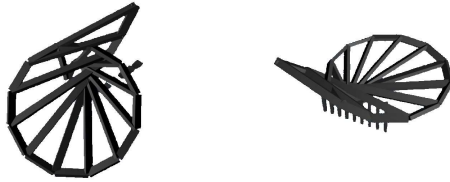

Figure 8   User 2: top and side views of final result.

This user also made the configuration change, explained above, from tournament to roulette-wheel selection. In an effort to increase diversity and explore larger areas of the search space, the mutation rate was increased for some runs from 0.02 to 0.2, but this had the undesired effect of largely destroying heritability. Offspring tended to look unrelated to their parents. This change was made after the results to be reported here were achieved.

In the fourth run, after about 10 generations, a good object was found, with a single support beam and a curved roof. Its main drawback was that the roof was too large for its single support—see Figure 9(a). Several generations later, an individual similar to Figure 9(b) appeared, probably through an independent line of descent. This individual of multiple support beams, and a version of the curved-roof individual, were selected as parents: in the next generation, an individual appeared which combined the two. This is an example of a successful and understandable crossover. Several generations later, a mutation modified the roof slightly (the curve has been compressed by variation of a numerical parameter) to give the individual shown in Figure 9(c). Feedback from the collaborators included the observation that the final shelter might have been designed by hand. With multiple supports (Figure 9(c)), it was considered to be potentially buildable, with the requirement that the supports be extended underground and secured with concrete.

User 4 was not involved with the development of the 3D software and instead regarded it as a black box system. No attempt was made to understand or change

(a) (b) (c)
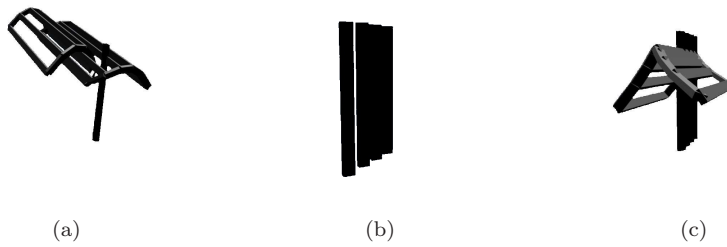
Figure 9 User 3: the first two individuals combined to give something similar to the last.

the configuration or representation details. The aim of this user was to achieve three appealing designs from a single run, without any preconceived target. Two of these results are shown in Figure 10. The left-hand individual is aesthetically appealing, with the polygon and butterfly shapes. The right-hand individual is easily buildable and would be more stable than many of the designs produced using this software. Its teepee/sea-shell shape was again seen as aesthetically appealing, and especially suitable to a rural or parkland setting, rather than an urban one. However it needs at least one post-evolutionary modification: one spoke needs to be removed to allow an entrance. Architectural and engineering feedback confirmed that the left-hand individual has a clear problem with structural integrity. The right-hand individual has the potential to scale to a multi-person shelter. Its curving and rotating features were similar to one of the designs produced by architecture students in response to the same design brief.
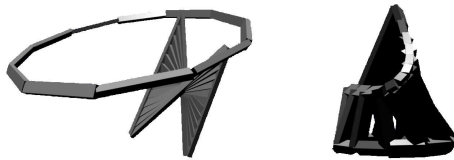
Figure 10 Final results from user 4.

5.5 *Discussion*

Although the tool was tested by users untrained in architecture and engineering, it is emphasised that this is not the (only) target audience for this technology. It is intended to be useful, in the longer term, to students, teachers, and practitioners of architecture and engineering. It is potentially useful in that it forces the conceptualization process to occur in three-dimensional space, which can often be inadvertently avoided when using traditional orthographic drawings. It is anticipated that the tool will undergo further testing within the architecture and engineering programmes at UCD in the coming academic year.

Several trends emerged from the design and evaluation process. Most users felt that selecting just one or two individuals per generation was the most effective strategy: this made it much more likely that the next generation would contain

an incremental improvement, rather than a large number of apparently unrelated designs.

Users also felt that the software was most useful as an exploratory tool, often producing results which would not have been considered or arrived at manually, rather than a method of searching for a known target. This is a common result in IEC. Known targets, whether preconceived or imagined as specific variations on existing individuals, are usually best produced by manual editing rather than evolutionary search.

Good individuals sometimes appeared early, suggesting that random search might be as effective as EC—however every user was sometimes able to improve designs incrementally through iteration, which is not possible with random search. This is a positive result.

The change in **population size** during runs was interesting. Initial generations often comprised only about 10 individuals, since several randomly-generated ones were invalid. Later generations were generally comprised of the full 15 individuals, as the user selected and bred from valid individuals and (often) selected against the over-complexity which can lead to invalid ones. This is however an implementation detail, and we can easily enfore the constraint that every individual produced is valid, and that the population size remains constant.

Recursion in the grammar is one advantage of the GE representation over fixed-length GA-style representations, in that GE can be quite open-ended, allowing objects of indefinite complexity. This is achieved through recursive grammar rules such as `<scene> ::= <shape> | <scene> + <shape>`. It is a necessary feature, since very often the designer does not know in advance how many components will be required. However, excessive complexity is not desirable, and recursion can easily lead to designs composed of far too many components arranged with little coherence. An example is given in Figure 11. When designs like this appear, they are generally not selected for crossover and mutation, and it is not too difficult to breed the corresponding genes out of the population. However, excessive complexity can reappear, and in IEC it is preferable to save the user as many fitness evaluations as possible by imposing constraints through the representation. We can argue, based on experience, that good individuals are always composed of 2 or 3 components (bearing in mind that each component may be composed of multiple, coherently-arranged beams), and express this constraint through the grammar. The rule given above is simply replaced with this one: `<scene> ::= <shape> + <shape> | <shape> + <shape> + <shape>`. Alternatively the constraint can be expressed as a configuration change: the maximum recursion depth can be made available as a numerical parameter in the design tool.

Figure 11   Excessive complexity arises through recursion.

The question of **connected and disconnected components** leads to a similar situation. Each `<shapes>` component is independent of the next, and so discon-

Figure 12   An individual together with variations produced by nodal and structural mutations. Nodal mutations tend to produce small, quantitative variations, whereas structural mutations can make larger, qualitative ones.

nected designs are common. Again, we can argue that the property of connectedness is known, *a priori*, to be desirable, and therefore that the user should not be required to waste fitness evaluations on imposing this constraint. Instead it can be imposed through the grammar. This will be discussed more in Section 6.2.

In IEC, the behaviour of the evolutionary operators appears to be particularly important. A central requirement is that parent individuals' features are not only inherited but they are *seen to be* inherited. That is, the user's perception of inheritance and control over it are indispensible. Crossover operators which produce children with no discernible similarity to their parents will tend to frustrate the user, who will feel that selections are not controlling evolution. A similar argument applies to mutation operators. In our system, operators often work conceptually, as in the example given in Figure 9. However, not every crossover or mutation was as successful and as understandable as the example given in Figure 9. It was not uncommon for a crossover or mutation to produce a new individual with little or no similarity to the original or originals. This is partly a consequence of the GE mapping process. One response to this problem is the idea of separating mutation into **structural** and **nodal** **mutations**. These two operators work by partitioning the array genome into two parts: those parts which, if mutated, would give rise to large, qualitative changes and smaller, quantitative changes in the phenotype, respectively. This idea has been explored in previous work and shown to be useful in controlling the destructive and creative effects of mutation (Byrne et al., 2009). Experiments here showed that allowing only nodal mutation tended to preserve the large-scale properties of the phenotype but change details, as illustrated in Figure 12. This suggests the possibility of allowing the user to specify which type of mutation is to be applied to which individual, relying on users' domain knowledge and intuition to identify the most useful *type of change* to be applied in each situation.

6   Conclusions and Future Work

6.1   Conclusions

The successes of this work consist in the harnessing of computer power as a tool in human creativity. The generative process allows the creation of unlimited amounts of novelty, to be shaped by an iterative procedure which does not require technical knowledge, but rather operates on the principle of *I know what I like*

when I see it. Domain knowledge can be expressed in a natural way through a formal grammar, and this leads to coherent, purposeful-looking designs. User-specific aesthetics and requirements can also be expressed through the grammar, and this can give better results, faster than using selection alone.

Overall this work-in-progress is considered to be a success, in that interesting, appealing, and surprising designs have been produced. The software, even at this early experimental stage, holds promise as an exploratory tool for students and practitioners for small scale projects. Much of the future usefulness depends on the level of complexity and the number of variables that can be incorporated into the evolutionary process while still maintaining the ease of use.

6.2   Future Work

The work presented here is ongoing and aims at eventually becoming a viable tool for architects to use in their design process. Several potentially useful new features have been identified. As discussed in Section 5.4, a context-sensitive mutation operator was used to improve the sense of similarity between related individuals. Similar context-sensitive crossover operators have been implemented in previous work, and could be applied to the 3D problem for the same purpose.

Another topic of future work is addressing connectivity between elements in the design. Currently, elements of the design have some limited connectivity, in that each element may be connected to only one other element. It may be beneficial to allow for longer chains of elements to be connected as well as elements that are completely unconnected. This could also be a possible solution to the problem of overly complex individuals discussed in Section 5.5.

An optional temporary storage (Dahlstedt, 2009) which can be used to re-seed evolution after stagnation, and to ensure that good individuals are not lost, making elitism unnecessary could be advantageous.

The grammars adopted in this study were fixed during the evolutionary process. With GE it is also possible to allow these grammars to evolve themselves O'Neill and Ryan (2004); O'Neill and Brabazon (2005); Hemberg et al. (2007a, 2008), and thus adaptively and automatically incorporate modules and features of the design that emerge over time. Another advantage of dynamic grammars is that if the users themsevles identify features or components of the design that are interesting then these could be directly added to the underlying grammar.

IEC is successful when it allows the user to evaluate populations as quickly as possible, i.e. addresses the fitness evaluation bottleneck (Biles, 1994). In particular, in graphical domains it is often possible to view an entire population in thumbnails, focussing in for careful viewing only of those individuals which are not obviously bad. This idea might be extended to allow a series of thumbnails representing local mutations to be displayed when mousing-over each individual.

There are many other enhancements and variations to the approach used here that may enable the user to better guide the search, including allowing user modification of the crossover and mutation rates and population size, mechanisms to handle multiple objectives (e.g., Deb et al. (2002); Purshouse and Fleming (2007)), and direct modification of the individuals such that they may be re-seeded back into the population.

Acknowledgements

References

Aoki, K. and Takagi, H. (1996). Interactive GA-based design support system for lighting design in 3-D computer graphics. *Trans. of IEICE*, 81:1601–1608.

Banzhaf, W. (1994). Genotype-phenotype-mapping and neutral variation—a case study in genetic programming. In *Proceedings of Parallel Problem Solving from Nature III*, volume LNCS 866, pages 322–332. Springer.

Banzhaf, W. (1997). Interactive evolution. In Back, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, chapter C2.9, pages 1–6. IOP Publishing Ltd. and Oxford University Press.

Bentley, P. J. (2000). Exploring component-based representations—the secret of creativity by evolution? In Parmee, I. C., editor, *Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000)*, pages 161–172, University of Plymouth.

Bentley, P. J. and O'Reilly, U.-M. (2001). Ten steps to make a perfect creative evolutionary design system. In *GECCO 2001 Workshop on Non-Routine Design with Evolutionary Systems*.

Biles, J. (1994). GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference*, pages 131–131. INTERNATIONAL COMPUTER MUSIC ASSOCIATION.

Brabazon, A. and O'Neill, M. (2006). *Biologically Inspired Algorithms for Financial Modelling*. Springer.

Byrne, J., O'Neill, M., McDermott, J., and Brabazon, A. (2009). Structural and nodal mutation in grammatical evolution. In *Proceedings of the GECCO-2009 Workshop on Non-routine Design with Evolutionary Systems*.

Cagan, J. and Vogel, C. M. (2001). *Creating Breakthrough Products: Innovation from Product Planning to Program Approval*. Prentice Hall.

Dahlstedt, P. (2009). On the role of temporary storage in interactive evolution. In Giacobini, M. et al., editors, *Applications of Evolutionary Computing: EvoWorkshops 2004*, number 5484 in LNCS, pages 478–487. Springer-Verlag.

Dawkins, R. (1986). *The Blind Watchmaker*. Longman Scientific and Technical, Harlow, England.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transaction on Evolutionary Computation*, 6(2):181–197.

Dempsey, I. (2007). *Grammatical Evolution in Dynamic Environments*. PhD thesis, University College Dublin, Ireland.

Dempsey, I., O'Neill, M., and Brabazon, A. (2009). *Foundations in Grammatical Evolution for Dynamic Environments*. Springer.

Dorris, N., Carnahan, B., Orsini, L., and Kuntz, L. (2004). Interactive evolutionary design of anthropomorphic symbols. In *CEC2004: Proceedings of the 2004 Congress on Evolutionary Computation*, volume 1.

Foreign Office Architects (2009). Foreign Office Architects. URL `http://www.f-o-a.net`. Accessed 13 May, 2009.

Funes, P. and Pollack, J. (1997). Computer evolution of buildable objects. In Husbands, P. and Inman Harvey, E., editors, *Proceedings of the Fourth European Conference on Artificial Life*, pages 358–367.

Gero, J. S., Louis, S. J., and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement. *AIEDAM*, 8:83–94.

Harper, R. and Blair, A. (2005). A structure preserving crossover in grammatical evolution. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume Volume 3, pages 2537–2544. IEEE Press.

Hemberg, E., Gilligan, C., O'Neill, M., and Brabazon, A. (2007a). A grammatical genetic programming approach to modularity in genetic algorithms. In Ebner, M. et al., editors, *EuroGP 2007: Proceedings of the 10th European Conference on Genetic Programming*, number 4445 in LNCS, Valencia, Spain. Springer.

Hemberg, E., O'Neill, M., and Brabazon, A. (2008). Grammatical bias and building blocks in meta-grammar grammatical evolution. In *In Proceedings of the IEEE World Congress on Computational Intelligence*, pages 3775–3782, Hong Kong. IEEE Press.

Hemberg, M. and O'Reilly, U.-M. (2004). Extending grammatical evolution to evolve digital surfaces with Genr8. In Keijzer, M., O'Reilly, U.-M., Lucas, S. M., Costa, E., and Soule, T., editors, *Genetic Programming 7th European Conference, EuroGP 2004, Proceedings*, volume 3003 of *LNCS*, pages 299–308, Coimbra, Portugal. Springer-Verlag.

Hemberg, M., O'Reilly, U.-M., Menges, A., Jonas, K., da Costa Goncalves, M., and Fuchs, S. (2007b). Genr8: Architect's experience using an emergent design tool. In Machado, P. and Romero, J., editors, *The Art of Artificial Evolution*. Springer-Verlag, Berlin.

Hicklin, J. (1986). Application of the genetic algorithm to automatic program generation. Master's thesis, University of Idaho, Moscow, ID.

Hornby, G. S. (2005). Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of GECCO '05*.

Keller, R. and Banzhaf, W. (1996). Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 116–122, Stanford University, CA, USA. MIT Press.

Kicinger, R., Arciszewski, T., and Jong, K. D. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers and Structures*, 83(23-24):1943 – 1978.

Koning, H. and Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and Planning B*, 8:295–323.

Koza, J. R. (2003). *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, Norwell, MA, USA.

Mayall, K. and Hall, G. B. (2005). Landscape grammar 1: spatial grammar theory and landscape planning. *Environment and Planning B: Planning and Design*, 32:895–920.

Mayall, K. and Hall, G. B. (2007). Landscape grammar 2: implementation. *Environment and Planning B: Planning and Design*, 34:28–49.

McDermott, J. (2008). *Evolutionary Computation Applied to the Control of Sound Synthesis*. PhD thesis, University of Limerick.

McKay, R. I. B., Nguyen, X. H., Whigham, P. A., and Shan, Y. (2005). Grammars in genetic programming: A brief review. In Kang, L., Cai, Z., and Yan, Y., editors, *Progress in Intelligence Computation and Intelligence: Proceedings of the International Symposium on Intelligence, Computation and Applications*, pages 3–18, Wuhan, PRC. China University of Geosciences Press.

O'Neill, M. (2001). *Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution*. PhD thesis, University of Limerick.

O'Neill, M. and Brabazon, A. (2005). mGGA: The meta-grammar genetic algorithm. In *Proceedings of the European Conference on Genetic Programming EuroGP 2005*, volume LNCS 3447, pages 311–320, Lausanne, Switzerland. Springer.

O'Neill, M. and Brabazon, A. (2006a). Grammatical differential evolution. In *Proceedings of IC-AI*, pages 231–236. CSREA Press.

O'Neill, M. and Brabazon, A. (2006b). Grammatical swarm: The generation of programs by social programming. *Natural Computing*, 5(4):443–462.

O'Neill, M. and Brabazon, A. (2008). Evolving a logo design using Lindenmayer systems, Postscript and grammatical evolution. In *IEEE Congress on Evolutionary Computation 2008*, pages 3788–3794, Hong Kong, China. IEEE Press.

O'Neill, M. and Brabazon, A. (2009). Recent patents in genetic programming. *Recent Patents on Computer Science*, 2(1):43–49.

O'Neill, M., Hemberg, E., Bartley, E., McDermott, J., and Brabazon, A. (2008). GEVA: Grammatical evolution in java. *SIGEVOlution*, 3(2):17–22.

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

O'Neill, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers.

O'Neill, M. and Ryan, C. (2004). Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In *Proceedings of the European Conference on Genetic Programming*, pages 139–148. Springer.

O'Neill, M., Ryan, C., Keijzer, M., and Cattolico, M. (2003). Crossover in grammatical evolution. *Genetic Programming and Evolvable Machines*, 4(1).

O'Reilly, U.-M. and Hemberg, M. (2007). Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2):163–186.

O'Reilly, U.-M. and Ramachandran, G. (1998). A preliminary investigation of evolution as a form design strategy. In Adami, C., Belew, R. K., Kitano, H., and Taylor, C. E., editors, *Proceedings of the Sixth International Conference on Artificial Life*, pages 443–447, University of California, Los Angeles. MIT Press.

Parmee, I. and Bonham, C. (2000). Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies. *AI EDAM*, 14(01):3–16.

Poli, R., McPhee, N., and Langdon, W. (2008). *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`.

Purshouse, R. and Fleming, P. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE Trans. Evolutionary Computation*, 11(6):770–784.

Ratle, A. and Sebag, M. (2000). Genetic programming and domain knowledge: Beyond the limitations of grammar-guided machine discovery. In *Proceedings of Parallel Problem Solving from Nature PPSN VI*, volume LNCS 1917, pages 211–220. Springer.

Reddin, J., McDermott, J., and O'Neill, M. (2009). Elevated pitch: Automated grammatical evolution of short compositions. In Giacobini, M. et al., editors, *Applications of Evolutionary Computing: EvoWorkshops 2004*, number 5484 in LNCS, pages 579–584. Springer-Verlag.

Ryan, C., Collins, J. J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In Banzhaf, W., Poli, R., Schoenauer, M., and Fogarty, T. C., editors, *Proceedings of the First European Workshop on Genetic Programming*, volume 1391, pages 83–95, Paris. Springer-Verlag.

Sims, K. (1991). Artificial evolution for computer graphics. In *SIGGRAPH '91: Proceedings of the 18th annual conference on computer graphics and interactive techniques*, pages 319–328, New York, NY, USA. ACM.

Stichting Blender Foundation (2009). Blender 3D. `http://www.blender.org/`. Last viewed 11 May 2009.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B*, 7:343–351.

Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proc. of the IEEE*, 89(9):1275–1296.

Vogel, C. M. and Cagan, J. (2005). *The Design of Things To Come.* Prentice Hall.

Whigham, P. (1996). *Grammatical Bias for Evolutionary Learning.* PhD thesis, University of New South Wales, Australian Defence Force Academy, Canberra, Australia.

Wong, M. and Leung, K. (2000). *Data Mining Using Grammar Based Genetic Programming and Applications.* Kluwer Academic Publishers.

Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). Instant architecture. *ACM Trans. Graph.*, 22(3):669–677.