GENR8 - A Design Tool for Surface Generation

Martin Hemberg
Chalmers University of Technology
412 96 Gothenburg
Sweden
f96mahe@dd.chalmers.se

Una-May O'Reilly
Artificial Intelligence Lab
Massachusetts Inst of Tech
Cambridge, MA 02139
unamay@ai.mit.edu

Peter Nordin
Department of Physical Resource Theory
Chalmers University of Technology
412 96 Gothenburg
Sweden
nordin@fy.chalmers.se

Abstract

GENR8 is an architect's design tool that generates surfaces. It is powerful and innovative because it fuses expressively powerful universes of growth languages with evolutionary search. Developed via the API of Alias|Wavefront's Maya, it combines 3D map L-systems, that are extended to an abstract physical environment, with Grammatical Evolution. GENR8 addresses key issues arising from exploiting evolutionary adaption within a creative interactive tool framework. EAs typically adapt 'off-line' but GENR8 is designed to sensitively accommodate the nature of the back and forth control exchange between user and tool during on-line evolutionary adaptation. It addresses how users may interrupt, intervene and then resume an EA tool. It also forgoes interactive subjective design evaluation for computationalized multi-criteria evaluation that permits wider search in shorter time spans.

1 INTRODUCTION

Evolutionary algorithms have traditionally been used to solve optimization problems. However people have tried to use them for creative purposes, for instance to generate artificial life forms or as a design aid. Lately they have been used for evolutionary art, helping the artist to create new forms by exploring a wide range of forms (Soddu, 2001). Bentley (Bentley, 2001) gives an overview and a description of evolutionary design projects.

The Emergent Design Group at MIT [2] unites architects from the School of Architecture and computer scientists from the Artificial Intelligence Lab. The architect members embrace a process of design that stresses emergence. In this process the complex aspects of an architectural scenario are studied and addressed with a bottom up methodology. Primitive level investigations pursue distinct realms such as morphology (ie form, shape), material, structure, program etc, and subsequently are integrated and non-linearly combined to shape the architectural experience and artifacts that respond to the scenario.

Creativity in form is one essential thread in the process. The architects desire computational tools that will stimulate their creativity and contribute to their creative output. In particular, they are intrigued by natural form and biological growth processes that underlie its formation. They find an organic quality to surfaces very compelling.

With this broad motivation in mind, we have designed GENR8. It is a surface modeling tool that simulates organic growth of surfaces in an environment. We found map L-systems to be suitable model for this kind of growth. However, we had to extend the map L-system model to make it work in 3D, and we have added environmental elements that influence and interact with growth. To search the universe of possible surfaces, GENR8 uses evolutionary computation. The details of GENR8's growth process and its evolutionary component will be described in Section 2.

Using Evolutionary Algorithms has two benefits; the first is that the parallel population based search gives us multiple solutions to the design problem. This is a feature that was requested by the designers; they do not want just one solution, but several alternatives on the same theme. Evolutionary algorithms also accommodate discovery within the extremely large universe of surfaces; it is selective and explorative yielding adaptation and discovery.

Yet, EAs also have short-comings. The fitness evaluation is a particularly tricky part for creative evolution-

ary systems. In GENR8 we forgo user inspection and ranking for an automated solution. Fitness evaluation will be discussed in Section 4.3.

Another issue with evolutionary design tool is control.Who has control, the tool or the user? The traditional way is to set up the evolutionary run, let it run to completion and then give the user access to the final output. However, architects desire more complete control of the creative process. They want to be to direct the system and hav it react to their changing desires at any point in the design process. In section 4.2, we discuss how user tool control is negotiated and implemented in GENR8.

If GENR8 is to have any practical value to a designer it must fit into the architects' tool box. Architects have a wide range of tools that can be used for different purposes and in different stages in the design process. In Section 4.1 we elaborate upon how GENR8 fits a niche within a more comprehensive design process.

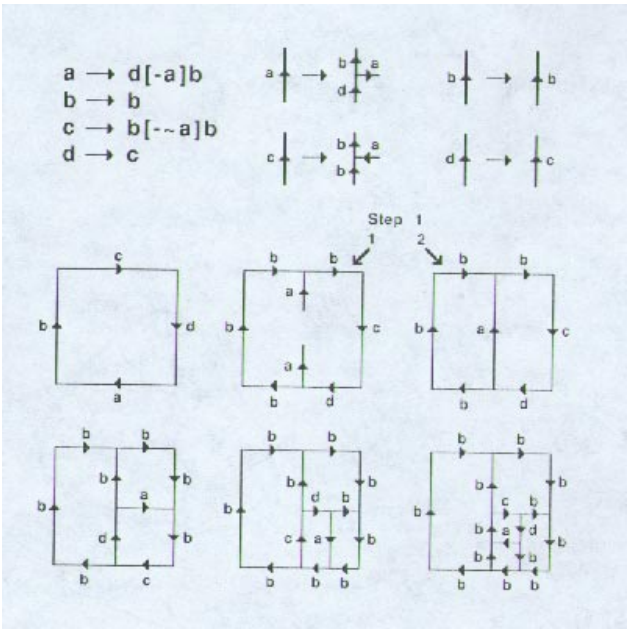

Figure 1: An example of a simple map L-system

2 GROWTH MODEL

Our goal is stated as a system that grows surfaces in an organic fashion. We use *map L-systems*, an extension of the more familiar L-systems as a basis for our growth model. To search the universe of possible grammars, we use the Grammatical Evolution EA. The EA relies on a BNF representation of the map L-systems, defining a universe of grammars. Grammatical Evolution, employs a BNF to map of a fixed length integer genome into an executable structure. GENR8 has two mapping processes, one that maps a genome to a grammar and a another that interprets the grammar and constructs a surface (phenotype). We will start by describing the second step, the growth model.

2.1 HEMBERG EXTENDED MAP L-SYSTEMS

L-systems were invented by biologist Aristid Lindenmayer in 1968 and they were first used to simulate the growth of plants. A description of L-systems can be found in (Prusinkiewicz 1989). One powerful aspect of L-systems is that it is a recursive growth model which means that the rules can be very succinctly expressed.

Map L-systems were originally invented as a model for cellular development. It is basically a method for rewriting planar graphs with cycles. An example can be found in Figure 1.

With a suitable graphical interpretation, the graphs can be used in a biological context to represent cellular structures. Whereas an L-system generates an arboreal structure, a map L-system generates graphs that can be interpreted as surfaces.

The original map L-system model is only defined for 2D and in order to make it work in 3D, we have made some additions to the model. We call them Hemberg Extended Map L-systems (HEMLS).

2.2 ENVIRONMENT

GENR8 has a very powerful environment that has great impact on the development of a surface. In the general case it is impossible to predict what a certain grammar will result in, unless you know what the environment looks like.

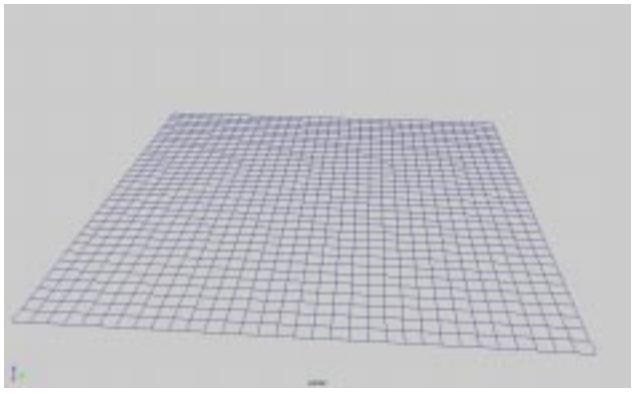

Figure 2: A surface grown in an empty environment

2.3 Forces

The GENR8 environment models forces. There are three types of forces; attractors, repellors and gravity, the user may place attractors and repellors in the surrounding space. Attractors and repellors work like magnets, and can be used to control the direction of the growth. They are situated in space and their effect depends on the relative location of the surface. In figure 3 the surface is generated by the same grammar as in Figure 2, but the five repellors shapes the final result.

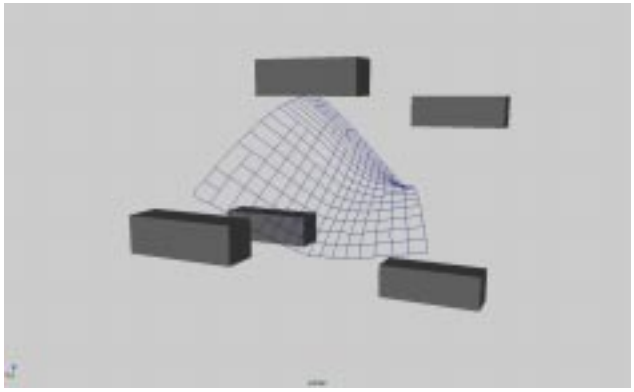

Figure 3: A surface grown in an environment with five repellors.

Gravity is a uniform force that has the same effect on all parts of the surface, regardless of its position (unlike attractors and repellors). The effect of gravity can be seen in figure 4, where we once again have the same grammar as in Figure 2.

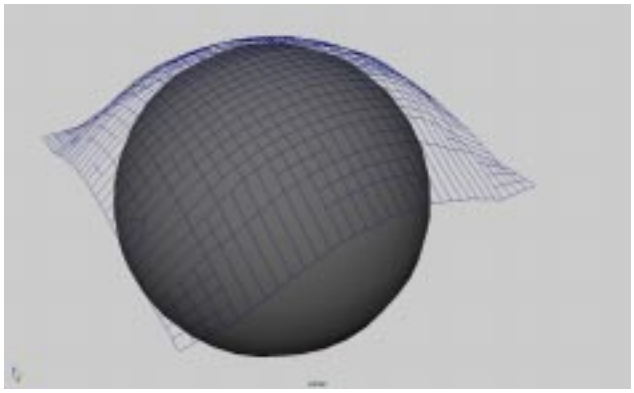

Figure 4: This surface is pulled downwards by gravity, but it is blocked by the sphere.

2.4 Boundaries

The user may draw arbitrary surfaces and volumes that act as boundaries. The walls can affect the growing surface in three different ways yielding different results. The effect of boundaries is illustrated in Figure 4 and 5.

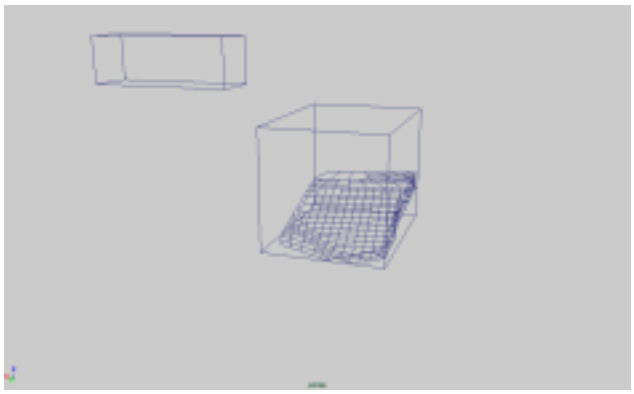

Figure 5: Surface inside a bounding box. The repellor in the upper left corner pushes the surface into the corner of the bounding box.

3 EVOLUTIONARY COMPONENT

To tell what kind of surface a HEMLS grammar will produce is a very hard task. Constructing grammars by hand is tedious and difficult and we can not expect a person without considerable expertise to accomplish this. In order to make HEMLS useful for designers who are interested in creating surfaces, we use an EA to generate and evaluate grammars.

3.1 GRAMMATICAL EVOLUTION

GENR8 employs an EA invented by O'Neill and Ryan (Ryan and O'Neill, 1998) called Grammatical Evolution (GE). GE is based on standard GA. It uses standard genetic operations on a fixed length vector of integers. These integers are then used to generate an executable structure from a BNF-specification of the language. This introduces an additional mapping that does not exist in traditional GAs.

Grammatical evolution provides genetic degeneracy. That is, there are multiple gene encodings that map to one decoding.With the first step of its two step genetic mapping process (BNF to Map L-system to scaffold), it allows GENR8 to provide users with different universes of HEMLS.

3.2 BNF GRAMMAR FOR HEMLS

To exploit GE, GENR8 operates with a Backus-Naur Form(BNF) definition of HEMLS. The BNF presented here is the most general that the system is able to parse.

A grammar is represented by a tuple {N, T, S, P}, where N is a set of non-terminals, T is a set of terminals, S is a start-symbol (it has to be a member of N) and P is a set of production rules that maps the elements of N to T. We have:

```
N = { L-System, Axiom, RewriteRule, Operator,
      Predecessor, Successor, Modifier,
      Condition, Segment, Constant }

T = { +, -, &, ^, \, /, ~, [, ], <, >, ->,
      Edge, i, If, Weight, Angle, Sync,
      BranchAngle }

S = { <L-System> }
```

The production rules are defined as

```
<L-System> ::= <Axiom> <RewriteRule>
               { <RewriteRule> } Angle
               Constant [ Sync ]
               [ BranchAngle ]

<Axiom> ::= <Segment> [ ~ ] + <Segment> [ ~ ]
            + <Segment> { [ ~ ] + <Segment> }

<RewriteRule> ::= <Predecessor> ->
                  <Successor> [ <Condition> ]
                  [ Weight Constant ]
                  { <Successor> [ <Condition>
                  ] [ Weight Constant ] }

<Successor> ::= { <Modifier> } <Segment>

<Predecessor> ::= <Segment> |
                  <Segment> ``<'' <Segment> |
                  <Segment> ``>'' <Segment> |
                  <Segment> ``<'' <Segment>
                  ``>'' <Segment>

<Modifier> ::= { <Segment> } |
               <Modifier> ``[`` <Successor>
               ``]'' <Modifier> |
               <Operator> <Modifier>

<Operator> ::= + |
               - |
               & |
               ^ |
               \ |
               / |
               ~

<Segment> ::= EdgeX |
              EdgeX_i |
              EdgeX_i+1 |
              EdgeX_i-1 |

<Condition> ::= If i < Constant |
                If i > Constant |
                If i = Constant

<Constant> ::= 0 | 1 | 2 | 3 | ...
```

The terminals `Angle`, `Sync` and `BranchAngle` control parameters for the growth that has been genotypically encoded rather than user-defined.

Since we start from a valid grammar and use the Grammatical Evolution technique, we are certain that we will always have a valid grammar. Combined with the model for generating surfaces we can be quite sure that we will have a valid result.

It is also possible to have probabilistic `RewriteRule`, which means that there are several possible `Successor` to a `Predecessor` and the choice is random (according to some predefined distribution).

For the GE we use 4 BNFs that are slightly more restricted than the BNF described above. The reason for this is to narrow down the search space and obtain interesting results faster. These BNFs will generate different universe of surfaces, giving the user greater control and flexibility. Apart from the default BNF there is one that creates symmetric surfaces, one that generates grammars with probabilistic rewrite rules and one that produces reversible grammars. The reversible grammars is a subset that we can map back to a genotype; this is useful if we want to let the EA explore variations of the grammar. A grammar generated by the default BNF may look like this:

```
Edge1 + Edge1 + Edge2 + Edge2 + Edge2 + Edge3
Edge1 > Edge1 -> & [ [ - - Edge1 ] + + Edge1
        ] ^ Edge3
Edge2 > Edge2 -> [ [ - - Edge0 ] + + Edge0 ]
        Edge2
Edge1 < Edge2 -> ~ [ [ - - Edge3 ] + + Edge3
        ] Edge0
Edge3 -> + [ [ - - Edge3 ] + + Edge3 ] -
        Edge1
Edge0 > Edge0 -> / Edge3 Edge3 \ Edge2
Angle 45
BranchAngle 75
```

The resulting surface can be seen in Figure 6.

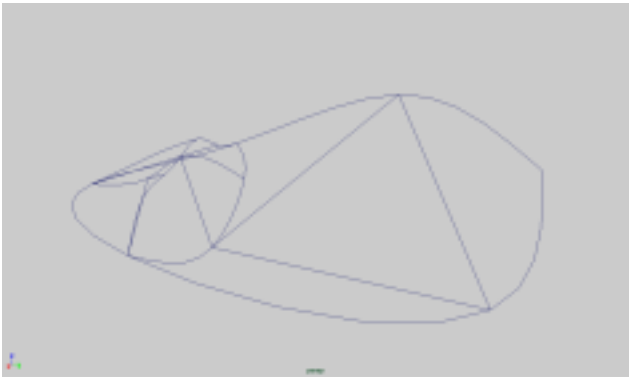

Figure 6: A surface that was evolved by the system from a population size of 15 after 20 generations.

3.3 MAPPING

The EA in GENR8 has a two step mapping from genotype to phenotype. First the genotype is mapped to an executable structure, a grammar. Then the grammar is interpreted and a surface is grown.

3.3.1 Genotype to grammar

Commencing from a start symbol, `S`, we read the genome to determine what production rule should be used. Standard to Ryan and O'Neil, the genevalues dictates the corresponding production rule. In the cases where a terminal or non-terminal in the production rule is optional or may occur multiple times, we use the genes to make the choice. In the case of multiple occurrences we use a method that is similar to an exponential probability distribution, so that there is no fixed upper limit to the number of occurrences of the terminal (or non-terminal). For instance, if we have a sequence of genes:

```
617 666 800 8
```

And we are about to expand the following expression:

```
{ <Segment> }
```

The brackets surrounding the non-terminal indicate that we are going to have an optional number of `<Segment>`. We use the genes to determine how many; by testing `617` mod $2 = 1$. This is greater than the current number of expansions from this node, so we add a `Edge` node. Next we test `666` mod $3 = 0$, which is less than the number of expansions, so we stop expanding this node. We continue by expanding the `<Segment>` terminal. We have four productions to choose from and the choice is made by taking `800` mod $4 = 0$. Finally we determine the type of the `Edge` by taking `8` mod $4 = 0$, where the modulo is the current number of edge types (in this example arbitrarily chosen to be 3) plus one. Thus we end up with `Edge0`, when the expansion is finished.

3.3.2 Grammar to phenotype

For the second step of the mapping, from grammar to surface, we use a variant of turtle graphics (Prusinkiewcz 1989). The grammar is interpreted as instructions for how the turtle should move and draw lines.

4 GENR8 AS A TOOL

Evolutionary Algorithms can be useful in a design software. They offer powerful search within the universe of possible designs. Designers have a plethora of software tools at their disposal. The tools have different strengths and it is neccesary that the designs are portable between the different tools. However, this is not enough to make it a useful tool. The EA component has to be integrated into the rest of the tool seamlessly so that it can be used in an intuitive way.

To make the GENR8 easily accessible for designers it has been implemented as a plug-in to an existing design tool. In this way it is easier to become familiar with it and, from a developing point of view, we get a lot of functionality for free. After consulting the designers in the Emergent Design Group, the choice of host software was Alias|Wavefront Maya. Many designers use this software and it has good support for writing plug-ins.

The EA component of GENR8 can be seen as a tool within a tool within a tool. At the top level, Maya is just one of many 3D-modeling software tools available to designers. GENR8 is just one of the many tools available when using Maya. Finally EAs is the tool used by GENR8 to create novel surfaces.

4.1 INTEGRATION INTO MAYA

The integration with Maya is seamless. GENR8 is implemented as a MEL-command (Maya Embedded scripting Language) and one uses it in the same way as any other feature in Maya. However, MEL is based on a command line interface and to make life easier for the designer, we have implemented a GUI.

From the GUI (and the command-line), you can set all the parameters for a run. The GUI is more user-friendly since it prevents the user from entering invalid

combinations. To make life even easier for the user, there are help files in HTML-format readily available.

Boundaries are set up using ordinary Maya surfaces. The user can draw arbitrary convex surfaces (concave surfaces usually work fine, but there are no guarantees) and they will be treated as walls by GENR8. Attractors and repellors can be placed using special-purpose commands.

The user may draw a curve and that curve will be used as a starting point for the growth (this will override the genotypically encoded `Axiom` which always is a regular polygon).

The surfaces are drawn in separate layers which makes it easy to toggle the visibility. This is very useful when inspecting the population. If the user wants to study an individual closer, there is a feature that allows one to re-grow that member and thus study it in closer detail. It is also possible to save both the grammar, the genome and the actual Maya surface.

4.2 INTERRUPTION, INTERVENTION AND RESUMPTION

The traditional way to use EAs in design tools is to have the user set up a run and then wait for the output. This can be very frustrating to a designer since it may alienate him or her from the process. We do not want to create a black box that just spits out a finished design; instead we are trying to make a tool that is *cooperates the designer*.

Our goal is to allow the designer to take an active role in the evolutionary process and to have a sense of control. An analogy that might be helpful is that of a car on cruise control. The car goes forward by itself but the driver is still in control of the steering and may hit the brakes, regaining full control. We have labeled this idea interruption, intervention and resumption (IIR). The user should be able to interrupt the EA at any time; an interesting design may be spotted or the whole process may need to be rescued. Next, the user should be able to intervene by changing parameters (eg mutation rate) and the environment (eg add an attractor). Finally it should be possible to resume the process from where it was interrupted.

The user can choose to view the entire population, or just one individual during the run. Relevant statistics and data are displayed in another window.

The user can guide evolution by setting the fitness values by hand. This will have effect in the ensuing tournament, but the next time the individual is evaluated it will still get a bad fitness value. Indirectly, the user can affect fitness by changing the environment and thus affecting the growth of a surface, possibly leading to a totally different result. The fitness function has several parameters (rewarding different features of the surface) and changing these parameters will alter the ranking of the individuals. Finally, one can make copies of an individual and insert those copies into the population.

It is also possible to insert another population (that has previously been saved to a file) into the existing population. This enables one to insert new genetic material in a controlled way.

In order to give the user even greater control, we have added a feature that maps a grammar back to a genotype. Thus a user can construct a grammar by hand or modify an evolved grammar and give that as input to the system. This user-defined genotype can be inserted in to the population as the starting point for further runs. The drawback with this feature is that the mapping from grammar to surface (phenotype) is very complex and it requires great expertise to anticipate what a particular grammar will result in.

4.3 DESIGN EVALUATION

A key issue for EAs is the fitness evaluation. For creative design this is obviously a very hard task, since it involves aestethic criterion rather than an objective goal function.

The most common approach is to determine fitness either by using a mathematical function to evaluate the design (and thus we are once again back to optimization) or fitness is directly determined by the user. Since it is obviously very hard to capture the nature of the outcome of a creative design process in a mathematical function, this approach has some severe limitations. If the user acts as fitness function, creativity is unhampered; but on the other hand, it is tedious and time consuming to evaluate every step of the evolutionary process.

Traditional Interactive Evolutionary Computation (IEC) (Takagi 2001) often runs into problems with human fatigue. Since the user has to evaluate the entire population between each generation. This limits the population size and the number of generations in practical use. To circumvent this problem , IEC tends to focus on getting faster convergence. GENR8 instead allows the user to express his or her preferences by setting the parameters for the fitness function.

As mentioned above, the fitness function is a linear combination of five independent fitness criteria that each evaluates different features of the design. These

criterion are size, smoothness, soft boundaries, subdivisions and symmetry.

The size criteria simply measures the extent of the surface in the x and y directions (it is assumed that the normal is oriented principally in the z direction). Smoothness is a measure of how rugged the surface is (in the z direction), both locally and globally. Soft boundaries is actually a fourth wall behaviour, the surface may grow through the boundary, but it incurs a fitness penalty as is does so. Subdivisions measure of how finely the surface is subdivided. The symmetry obviously is measure of how symmetric the surface is with respect to the x and y axises.

The concept of a fitness function with multiple components serves the designer well. It is easy to emphasize the features that one is most interested in and it gives the system some of the human abilty to be responsive to several criteria at the same time. The different (independent) parameters can be used to express multi-level, non-linear and possibly conflicting goals of a designer. Recall, the weight and parameters of any criterion can be changes at any time during a run.

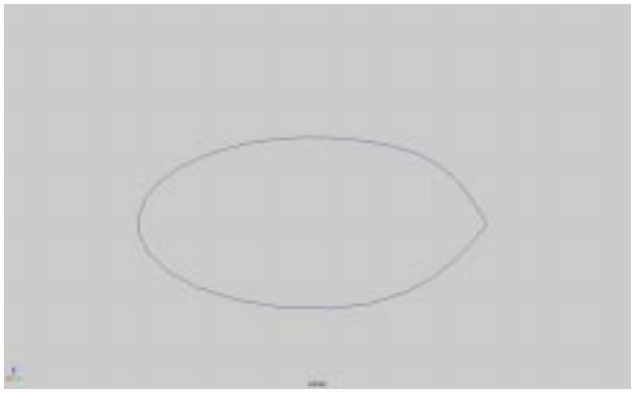

Figure 7: A randomized starting individual.

Figure 7 shows the best individual in a (randomized) starting population of size 50. When the parameters of the fitness function have their default values. If we then change the parameters to increase the number of subdivisions, after 5 generations, we get the result shown in Figure 8. If we wish to then make the surface more rugged rather than flat, we change the smoothness criteria. The result can be seen in Figure 9.

5 FUTURE WORK

GENR8 creates surfaces in 3D space and it would certainly be interesting to be able to view and evaluate the surfaces in a 3D environment rather than on a 2D monitor. Thus we would like to extend GENR8 through the use of some Virtual Reality technology.

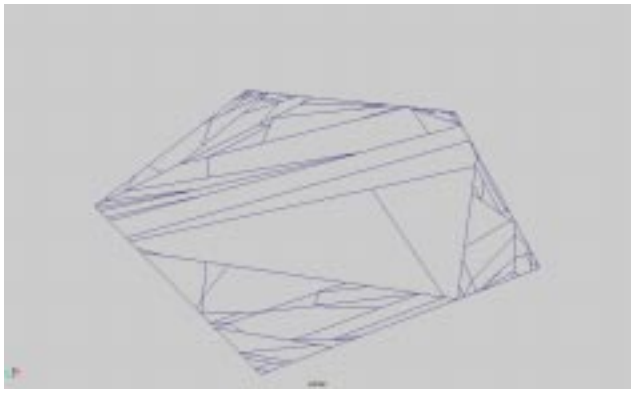

Figure 8: A subdivided surface.

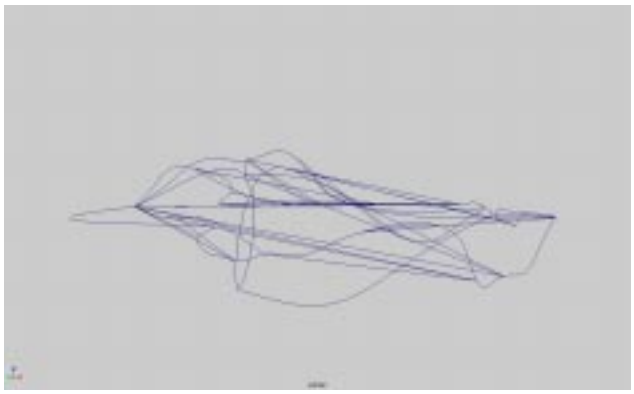

Figure 9: A rugged surface.

Right now, GENR8 creates surfaces, we would like to extend it so that we can grow solid designs as well. In the next incarnation we could use a solid modeller instead of Maya (it can only model surfaces). Solid modeling permits one to analyze the structural and material properties of the designs and incorporate this in the fitness evaluation. This could, for instance, be done using some finite element method.

Another interesting feature would be a machine learning system that observes the users behaviour and deduces the users preferences. This would lead to even more efficient and personalized fitness evaluation.

6 CONCLUSION

GENR8 makes EAs appear well suited for creative design by teaming them up with a powerful growth model. But it does so by placing strong emphasis on the user's perspective and adapting the EA to accomodate this.

7 ACKNOWLEDGEMENTS

The authors would like to thank Simon Greenwold, Peter Testa, Devyn Weiser and Janet Fan of the Emergent Design Group for their help with developing GENR8.

References

[1] Peter Bentley, *Aspects of Evolutionary Design by Computers*, www.cs.ucl.ac.uk/staff/P.Bentley/wc3paper.html, 2001.

[2] Emergent Design Group, web.mit.edu/arch/edg/

[3] Przemyslaw Prusinkiewicz, *Lindenmayer systems, fractals and plants*, Springer-Verlag, New York, 1989.

[4] Conor Ryan, Michael O'Neill, *Grammatical Evolution: A Steady State approach*, Proceedings of the Second International Workshop on Frontiers in Evolutionary Algorithms 1998, pp. 419-423.

[5] Tomoya Sato, Masafumi Hagiwara, *IDSET: Interactive Design System Using Evolutionary Techniques*, Computer-Aided Design 33, 2001, 367-377.

[6] Celestino Soddu, *Generative Art*, www.celestinosoddu.com, 2001.

[7] Hideyuki Takagi, *Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation*, Paper draft for proceedings of the IEEE that will appear in 2001 summer.

[8] Peter Testa, Una-May O'Reilly, Simon Greenwold, *Agency GP: The Architecture of Emergent Organizations*, Eternity, Infinity and Virtuality, ACADIA 2000 Annual Meeting Proceedings, Washington DC, in press.