

# ECSS – Evolutionary Component Surface System

## A tool for architectural form generation and form optimization

Katrin Jonas  
University College London  
Wates House 22 Gordon Street  
WC1H 0QB London UK  
+44(0)7834468980

katrin.jonas@ucl.ac.uk

Martin Hemberg  
Imperial College London  
Exhibition Road  
SW7 2AZ London UK  
+44(0)2075945169

martin.hemberg@imperial.ac.uk

### ABSTRACT

In this paper we describe the Evolutionary Component Surface System (ECSS). It is a software tool for generating, analyzing and optimizing structural component surfaces. The surfaces are created from a limited set of architectural parts. We can create a large number of different surface articulations using simple rules of connectivity. The generated structures are self supporting envelopes. Through their articulation the surfaces create inhabitable space and where they reach the ground they define their own points of support. The form-generating process is complemented with an evolutionary algorithm to help search the space of possible outcomes. An important aspect of the tool is that design as well as structural and fabrication aspects have been considered from the outset.

### Categories and Subject Descriptors

D.3.3 [Programming Languages]: Mel, Maya embedded language.

### General Terms

Algorithms, Design, Experimentation.

### Keywords

Evolutionary Design, Structural Optimization, Fabrication Optimization, Spatial Optimization.

## 1. INTRODUCTION

The advent of powerful computers provides novel and interesting opportunities for designers and design makers. Today, computer aided design (CAD) is extensively used by architects as a design tool. However we believe that only a fraction of the computers' potential is exploited. This project is part of a research effort aimed at making better use of available computing resources.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '06, July 8–12, 2004, Seattle, WA, USA.

Copyright 2006 ACM 1-59593-186-4/06/0007...\$5.00.

To achieve this goal, new ways of thinking about the role of the computer in the design process is required. Our intention is to develop algorithms that support new design conceptions and design making.

In this project we combine algorithmic growth with an optimization technique to explore the range of surfaces created from simple repetitive parts. The work is inspired by the concepts of complex adaptive systems and is based on the notions of Emergence [1] and Artificial Evolution [2]. These concepts are combined to create a digital design tool which can generate as well as optimize form. The project aspires to shorten the path from the conception of a design to its production. We are able to incorporate spatial, aesthetical and structural aspects within the same design composition process through the use of appropriate fitness criteria. The course of design conception and design making is understood as an iterative procedure where different interests and expertise need to interact. A software tool has the ability to programmatically combine different design considerations in a semi-automated process controlled by the user.

Applying ideas from complex systems to architecture is a relatively new one. An early pioneer in the field was John Frazer who started working on these ideas as early as in the 1960s [3]. He has also advocated the concept of creative design tools [4]. Developing software design tools based on the principles of complex systems was pursued by the Emergent Design Group (EDG) at MIT. Between 1997 and 2001, the group produced a number of different generative design tools. These tools emphasize interaction with the user and the idea of presenting multiple solutions to a given design problem. The tools by the EDG often provide visually compelling output as they employ advanced growth algorithms. However, their main drawback is that they are not informed by physical parameters. Thus, the user faces a significant challenge trying to understand the structural and material aspects of the available parameters [5]. Coates et al have also investigated the combination of Lindenmayer Systems with genetic Programming, which allowed them to produce recursively defined three-dimensional objects [6]. They stress that the use of evolutionary Algorithms in their experiments facilitate form generation not form optimization. An interesting example of a larger project based on self-organization is the Groningen Twister [7]. In this project, the columns in an underground bike storage are arranged using a self-organizing clustering algorithm. The placement of the columns is influenced by the structural requirements as well as the functional aspects of the space.

## 2. THE TOOL

The tool consists of two connected methods which we label engines. First there is the growth engine which is responsible for selecting and placing components and thereby combining them to form articulated surfaces. The growth engine has the capability to create complex and diverse structures from a small set of simple local geometries which are connected under simple rules. The second mechanism is the evolutionary engine. It helps the user to search the universe of possible surfaces for efficient solutions to a design problem. ECSS has been implemented as a script for Maya. This makes it easy to incorporate the evolutionary algorithm into a design process and intuitive for new users.

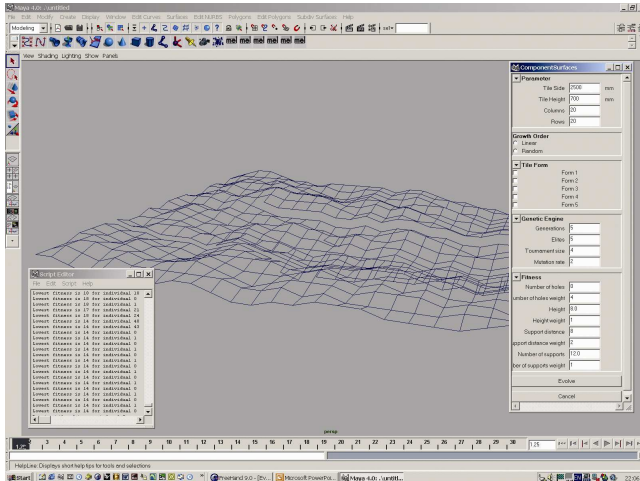


Figure 1. Screen shot in Maya, which showing a generated surface, the fitness score output window on the left side and the Graphical User Interface (GUI) on the right. In the GUI the user can specify the settings and choose the weights and target values for the fitness criteria.

### 2.1 The Growth Engine

The growth engine consists of a set of components and a set of rules that govern the interactions of the components based only on local information. By combining a large number of tiles, large articulated surface geometries may be created. An important criterion when developing the tool was to keep the number of components small and their geometries simple. The set of components in the ECSS consists of five basic forms which were derived from a square. By a controlled displacement of their vertices, we attain four variations of the original square. The components are shown in Figure 2. Each component except for the square can be rotated in 4 different ways. Thus, we end up with 17 different tiles grouped into five families.



Figure 2. The five initial components, which are derived from the ordinary square (to the left). The black lines illustrate the displacement of the vertices in reference to the original square. The controlled displacement ensures that the parts are sharing features so they can connect to each other and at the

same time they have distinct features to allow for differentiation in the overall surface.

The similarity of the components makes it possible to connect them in various ways. The rules for connecting the tiles are derived from the geometry of the component: two components can be connected if their edges have the same length and if they are inclined their inclination has to have the same direction (see Figure 3).



Figure 3. The five components with possible neighboring components overlaid.

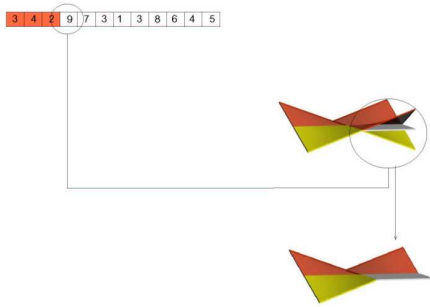
Another important rule is the order in which the tiles are added to the structure. In the basic setup, the tiles are added in a predefined linear fashion to form a square or rectangular envelope. As an alternative, the tile choice of where to put the next tile can be genetically encoded.

We wish to emphasize that the growth engine can be used independently and that it is a powerful form generating tool. However we have incorporated the second method, the algorithmic search and optimization to help the user explore the possible forms.

### 2.2 The Evolutionary Engine

Evolutionary algorithms are metaheuristic, population based, randomized search algorithms. The most common type of EA is the genetic algorithm (GA) [2]. EAs are inspired by natural evolution and they maintain a population of candidate solutions. They are frequently used for optimization problems in engineering and they are about to emerge in building applications. However, it remains an open question how they can be used to best effect. The number of possible surface outcomes in our surface generating system is excessively high. In fact, the number of possible surfaces grows combinatorial, making it intractable even for a very fast computer to investigate every possible surface consisting of a given number of tiles. By using a GA, we may efficiently navigate the space of potential solutions.

When applying a GA to a design task, the choice of how to encode the genome is very important. That is, how does one interpret the array of integers as a surface? Hornby and Pollack recommend using a grammar based approach as that will allow for an efficient programming of the genotype [8]. The solution in the ECSS is somewhere in between a direct representation and the grammatical encoding. The genome represents rules, but the choice of rule is determined by the evaluation of the phenotype so far. Thus, the representation is inefficient in the sense that there is a one to one mapping between the phenotype and the genotype. However, we do not encode the type of tile directly in the genome. Instead the number determines how to choose tiles from the subset of allowed tiles thus ensuring that we will always create valid individuals. The procedure for decoding the genome is illustrated in Figure 4.



**Figure 4. Mapping of the genome into a surface. The genome dictates which of the possible fitting components is chosen to fill the next position.**

### 2.3 Fitness Evaluation and User Control

The main problem when applying EAs to architectural design problems is to define a fitness function. As there is no generic method for evaluating fitness in architectural applications, this remains an open question. Romero et al suggest four different approaches: a) interactive evolutionary computation (IEC), b) learning user preferences, c) a rule based system and d) co-evolving critics [9]. The most popular of those four is IEC which bypasses the problem of defining a fitness function by having the user rank the members of each generation. This resolves the issue of defining fitness algorithmically but introduces another problem namely human fatigue. A user has a limited scope to objectively evaluate fitness in cases where the number of necessary considerations is high. Consequently the population sizes as well as the numbers of generations must be kept low. This restricts the possibility to find original designs [10]. The second approach is based on the idea of using a machine learning system, such as a neural network to learn the user's preferences. This method does not work very well in practice since there are too many parameters to learn from a small training set for the automated system. The third idea is to encode the user's knowledge about good designs as a set of rules. This approach works well for specific niche-problems but is hard to generalize.

Our approach is closest to the rule-based scheme, but an important feature is that the rules are parametric. That is, each rule considers a particular feature, such as the maximum height of the envelope. The target value for this parameter as well as its relative importance is defined by the user. To parameterize the rules gives the tool greater flexibility and allows for a wider range of applications. Moreover, we have retained the traditional IEC features, giving the user the possibility to micro-manage the selection.

In the ECSS there are three types of evaluation: a manual and two automated assessments. The manual evaluation allows the user to set the fitness score, giving the user the possibility to micro-manage the selection. However, in practice the parameterized fitness function is usually sufficient as it allows for significantly larger populations and longer runs. Second, there are the two

automated evaluations: the assessment of geometrical features and the assessment of structural behavior. The assessment of geometrical features through defined fitness functions is informed by spatial, aesthetic and structural considerations and takes place within the tool (see Table 1). This part of the automated evaluation is fast and occurs at every generation. After both of the described evaluation cycles the user can decide to continue under the same or altered parameters and fitness criteria.

The assessment of the structural behavior takes place in an interlinked finite element program (Ansys). This evaluation does not automatically take place during generation since it is computationally more costly. Typically, the user will run the generating process for several generations using only the manual and the geometrical evaluation. When the output is considered good enough, a set number of the fittest surfaces are read into the finite element analysis loop.

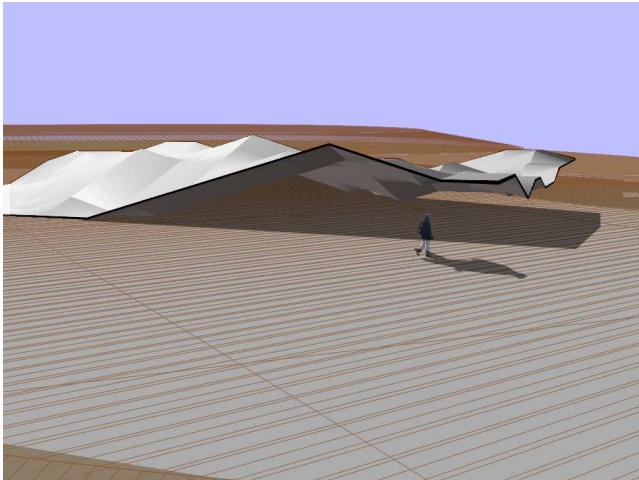
Ansys is started with the loaded script and the program runs through a number of automated steps. Material properties, thickness and size of a single component, represented as a mesh, is assigned. The components are connected to create a continuous area as we recognize the structures as shells. The lowest points along the z-axis of the surface are identified as support points. Under the set physical conditions Ansys calculates the structural behaviors, stress and the deflection, of each surface. Ansys outputs the location of the supporting points and the forces which they are subjected to. It also identifies the areas with the highest deformation. The fitness score is calculated and the score values are exported into Maya as text file.

In contrast to the first part of evaluation, the assessment criteria in Ansys are predefined, the user cannot choose within a given range. Again, the evaluation can lead to the selection of a hybrid geometry or a part of it or is followed by continuing the generating process under the same or altered parameters and fitness criteria.

The parametric fitness function of the ECSS is developed with the architectural application in mind to create self supportive envelopes (see Figure 5.). Through their articulation the surfaces define inhabitable space as well as that they are creating their own points of support.

**Table 1. The geometrical fitness criteria**

Number of holes	Number of holes which occur when no component fits into a place
Height	Highest point along the z-axis the surfaces should reach
Number of supports	Lowest points of the surface
Support distance	The desired minimum distance between lowest points



**Figure 5. Example of an architectural envelope.**

The choice of geometrical criteria is informed by aesthetical as well as spatial and structural considerations. For example, the distance and number of supports is structurally as well as a spatially important. The value range of the distance is informed by the size and material definition of the components. The current default is to use GFK sandwich panels each with a size of 2500mm\*2500mm\*84mm in plan projection. The size is determined by the sandwich panel fabrication technique. Computer controlled milling machines usually have a milling bed with the maximum capacity size of 2500\*2500mm which is also the maximum size for transportation. Through a number of Ansys tests we learned how far the surface created from these panels can span before they have to reach the ground again or being additionally supported. The maximum span informs the default value within which the user can choose.

### 3. CONCLUSION

We believe that generative growth algorithms provide exciting and novel opportunities for designers, engineers and fabricators. Different interests are no longer conflicting in an initial design proposal but are involved in the preliminary conception of it. Understanding, translating and tuning of the necessary conditions which a designer has to consider allows the interaction between those parameters to necessitate the design. They allow us to approach a design briefing without a preconception of form. The population based nature of the evolutionary algorithms provides us with multiple solutions to a given task which opens for new opportunities. Furthermore, computational power enables us to explore the possible range stemming from the combination of a set of simple parts. We are also able to represent constraints from material properties and production methods in the digital model. Particular to this project is that the fitness criteria consider aesthetic as well as engineering aspects thereby allowing for a programmatic form finding. The ECSS is still a prototype but nevertheless it is a demonstration of a novel type of device for designers. It is aimed at supporting an evolving understanding and interest in communicating design aspects and to parameterize design projects.

### 4. ACKNOWLEDGEMENT

The authors would like to thank Michael Hensel, Achim Menges and Mike Weinstock for many valuable and inspirational discussions. We also thank Matthias Michel with whom we discussed and developed the automated structural fitness evaluation.

### 5. REFERENCES

- [1] Holland, J. *Emergence: From Chaos to Order*, Oxford University Press, 1998.
- [2] Mitchell, M. *An Introduction to Genetic Algorithm*, MIT Press, 1996.
- [3] Frazer, J. *An Evolutionary Architecture*, Architectural Association Publications, 1995.
- [4] Jansen, P., Frazer, J. and Ming-Xi, T. *Evolutionary design systems and generative processes*, Applied Intelligence, 16, 2002, 119-128.
- [5] Hemberg, M. and O'Reilly UM. *Extending Grammatical Evolution to Evolve Digital Surfaces with Genr8*. European Conference On Genetic Programming (Coimbra, Portugal, 2004).
- [6] Broughton, T., Coates P. and Jackson, H. *Exploring 3D Design Worlds Using Lindenmayer Systems and Genetic Programming*. In *Evolutionary Design by Computers*, Morgan Kaufmann, 1999
- [7] Scheurer, F., *The Groningen Twister – An experiment in applied generative design*. In *Generative Art* (Milan, Italy, 2003)
- [8] Hornby, G. and Pollack JB. *The Advantages of Generative Grammatical Encodings for Physical Design*. Congress of Evolutionary Computation, 2001
- [9] Machado, P., Romero, J., Manaris, B., Santos, A. and Cardoso, A., *Power to the Critics – A Framework for the Development of Artificial Art Critics*. European Conference On Genetic Programming (Coimbra, Portugal, 2004).
- [10] Lutton, E., Cayla, E and Chapuis, J. *ArtiE-Fract: The Artist's Viewpoint*, 2003. European Conference On Genetic Programming (Essex, UK, 2003).