The use of Genetic Programming for applications in the field of spatial composition

Paul S. Coates AA Dipl & Luit Hazarika MSc B.Arch
CECA (Centre for Environment & Computing in Architecture)
University of East London, Stratford London UK
P.S.Coates@uel.ac.uk

1 Abstract

Architectural design teaching using computers has been a preoccupation of CECA since 1991. All design tutors provide their students with a set of models and ways to form, and we have explored a set of approaches including cellular automata, genetic programming , agent based modelling and shape grammars as additional tools with which to explore architectural (and architectonic) ideas.

This paper discusses the use of genetic programming (G.P.) for applications in the field of spatial composition. CECA has been developing the use of Genetic Programming for some time (see references) and has covered the evolution of L-Systems production rules , and the evolution of generative grammars of form . The G.P. was used to generate three-dimensional spatial forms from a set of geometrical structures. The approach uses genetic programming with a Genetic Library (G.Lib). G.P. provides a way to genetically breed a computer program to solve a problem. G.Lib. enables genetic programming to define potentially useful subroutines dynamically during a run .

In this paper we will report on new work using a range of different morphologies (boolean operations, surface operations and grammars of style) and describe the use of objective functions (natural selection) and the "eyeball test" (artificial selection) as ways of controlling and exploring the design spaces thus far defined.

2.Using Genetic Programming as a Generative Grammar of Form.

By a grammar we mean (after Stiny, Mitchell [1]), a set of initial conditions, a lexicon of primitive objects, and a syntax of transformations on those objects. In these experiments the grammars are on the one hand a "personal grammar" and on the other a pair of canonical architectural objects of wildly differing scales namely Le Corbusiers "DomIno House" and the Sear's Tower of Chicago.

We know what the initial grammar produces, when the simplest sentence produces the most basic design. GP allows the parallel exploration of the design worlds defined by the initial axioms and productions. Whether this is likely to be interesting depends entirely on the initial grammar. A badly chosen set of axioms and productions may lead to small design worlds. With a well chosen grammar, leading to a large number of non trivial design worlds, the likelihood of finding a suitable candidate as the solution to a properly posed problem increases.

Automated shape grammars have been reported, and the SEED project (Akin, 1997[3]) has developed automated shape grammar algorithms using prolog like syntax, with goal driven logic in the form of rule sets of valid relationships. In recent papers (Coates and Broughton,1997[2][13]) we are turning this procedure around, providing no rules, but, by evolving productions, allow for the emergence of grammatical objects by selection.

Genetic programming (Koza 92[4]) is a method of adapting the principles of evolutionary programming (such as Genetic algorithms, classifier systems and simulated annealing) to the evolution of programs. Koza's contribution was to propose that rather than representing genetic material (the allelles) of a phenotype as a coded string, they could be represented as a branching tree structure of nested functions. In the original examples the functions were arithmetic and mathematical, and the genotype was evaluated to make a numerical result.

In these examples the functions are geometric manipulations of form, and the genotype is evaluated to produce 3D objects. In particular, the functions are Autolisp functions that call Autocad 14 CAD operations. By using both artificial selection and natural selection.the GP system evolves 3D objects made up of successful collections of a small sub set of all possible trees of functionsof depth n.

The way in which such trees of cad functions is organised is based on embedding and recursion, which are the natural way in which Lisp datastructures and programs are created (in fact there is no conceptual difference between a list datastructure and a program, one is the other), and there is a parallel with phrase structure grammars (chomsky 65[5]) which define natural language in similar ways. GeneratingAutimatically Defined Functions (ADFs) could be seen as a method of isolating useful sub clauses in the evolving language.

It is this close connection between the representations that leads us to talk of design grammars, with we hope the added bonus that a computational model allows automatic exploration of design spaces defined using evolutionary algorithms.

3.The Solid Breeding Experiment

3.1 Breeding the AutocadSolid Primitives

Taking the notion (After Mitchell[6]) of a design grammar using a set of three-dimensional primitives and operators as inspiration, an AutoLlisp routine was developed which would be able to explore the design world that such an arrangement would have to offer.

3.2 The AutoCad Solid Primitives

The terminal set is composed of the basic three-dimensional Autocad primitives.

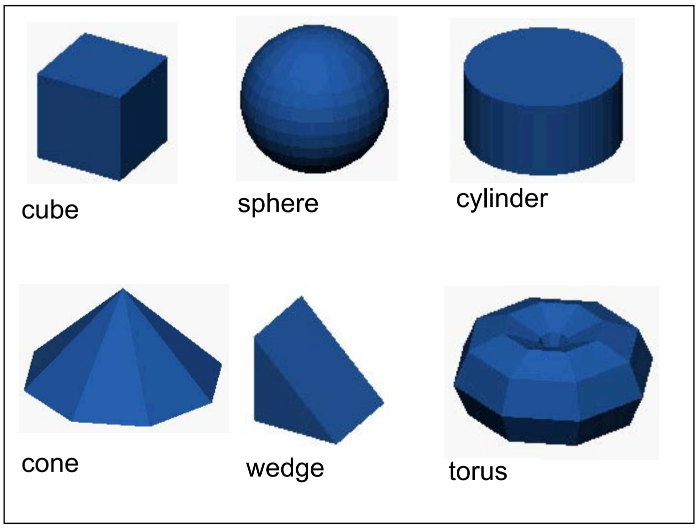


Figure 3.0
Set of six Autocad solid primitives.

3.3 The Function Set

The function set is composed of the various transformations that are possible to carry out on three-dimensional solid objects in AutoCad namely:

Move Transformations

The move series of transformations comprised of:
Moving 1 unit in all the orthogonal directions.

left ;right;forward ;back ;up;down

Copy Transformations

The copy series of transformations as with the move transformation comprised of:
Copying the object 1 unit in all the orthogonal directions.

left ;right;forward ;back ;up;down

Scale Transformations

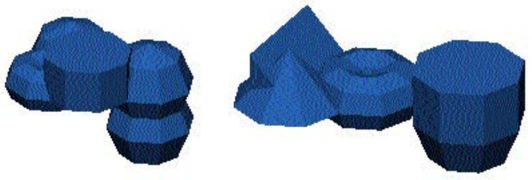
The scale transformations comprised of:
Scaling the object by a factor of 2 (double the original size).
Mirror Transformations
The mirror series of transformations comprised of:
Mirroring the object to the left and to the right.
Rotate Transformations
The rotate series of transformations comprised of:

rotate 45degrees left
rotate 45degrees right
rotate 45degrees front
rotate 45degrees back
Boolean Transformations

The boolean series of transformations unioned, subtracted or intersected the two objects which were picked to be in the selection set.

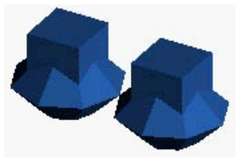
3.4 Results of Transformations on Primitive Set.

Figure 3.1 Results of the Move



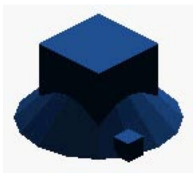
3rd generation individual 4th generation individual

Figure 3.2 Results of the Mirror



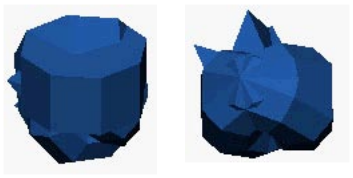
5th generation individual

Figure 3.3 Results of the Scale



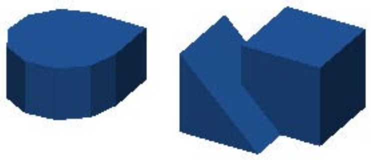
6th generation individual

Figure 3.4 Results of the Rotate



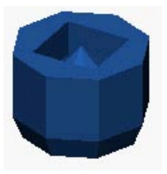
4th generation individual 5th generation individual

Figure 3.5: Results of the Union



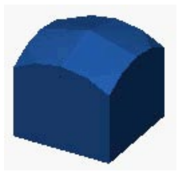
5th generation individual 7th generation individual

Figure 3.6: Results of the Subtract



5th generation individual

Figure 3.7 Results of the Intersect



7th generation individual

3.5 Results of all the Transformations and all the Terminals in one routine.

The next experiment involved combining all of the above operations into one routine, to see just what kind of compositions would be possible and what kind of individuals would be created if the individuals were allowed to breed for several generations. Figure 3.8 shows an isometric view of the individuals generated from running such an experiment. 8 generations were run and this time the whole array from generation 1 to generation 8 is shown, with the first generation in the bottom left corner. One can see that by the time you get to the fifth generation quite complex individuals are being generated.

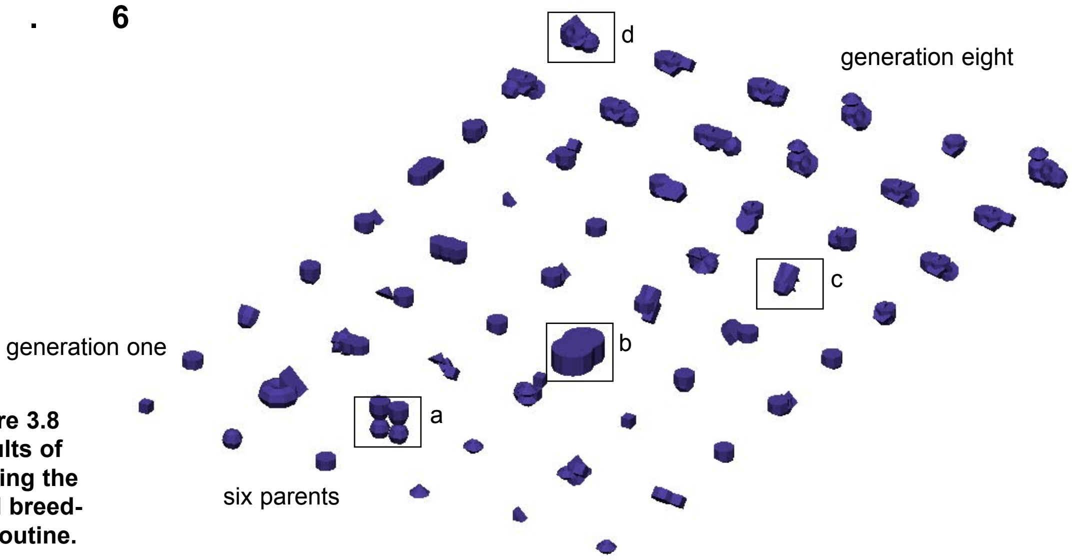


Figure 3.8 Results of running the solid breeding routine.

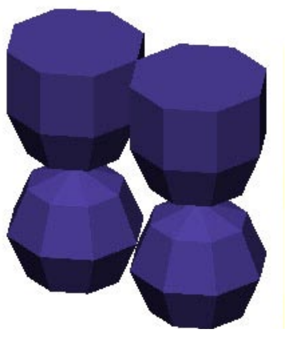


Figure 3.9

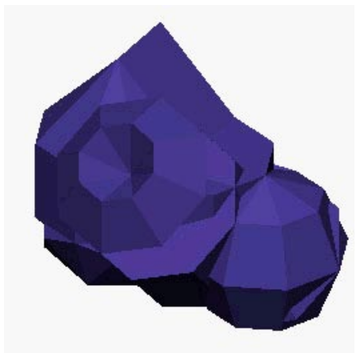


Figure 3.12

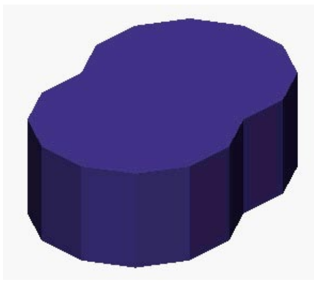


Figure 3.10

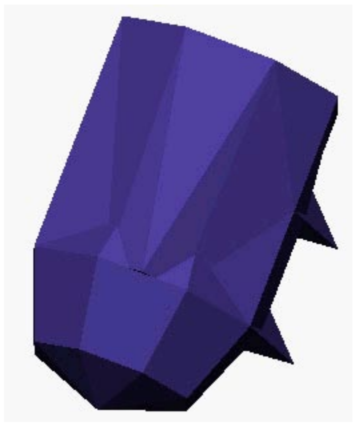


Figure 3.11

It is possible to suggest which operations took place on which terminals when looking at an individual. The following four individuals were picked from the field of individuals to show how the six initial primitive parents can produce quite complex offspring.

Individual A (figure 3.9)

This individual is most likely to be the result of a mirror and union operation on a cylinder and sphere.

Individual B (figure 3.10)

B is the result of a union and scale operation as it is twice as large as other individuals in the field.

Individual C (figure 3.11)

C is the result of various unions but it's most noticeable characteristic is the tilt which is the result of the rotation operation.

Individual D (figure 3.12)

D is one of the more complex individuals where several boolean operations have taken place along with rotations and movements.

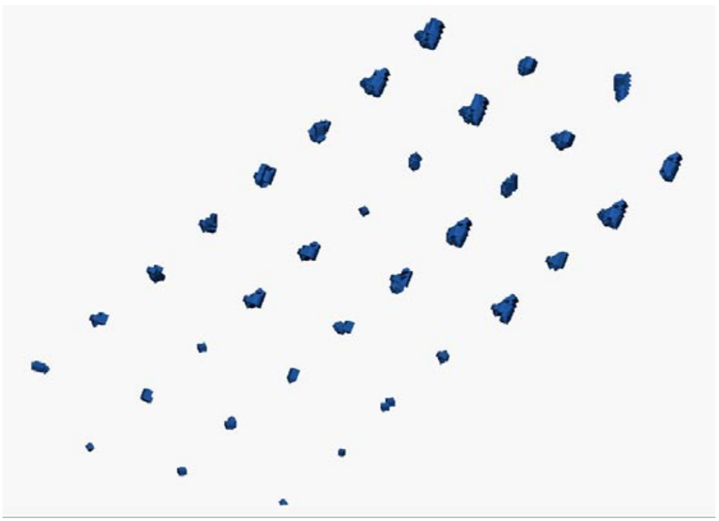


Figure 3.13 Array of individuals

3.7 Selecting an Individual using Aesthetic Criteria (The Eyeball Test)

Figure 3.13 above shows the individuals created after running the solid breeding program using the 'eyeball test' to decide which individuals become parents or are mutated to create the next generation. In this instance individuals were chosen for their complexity shape and general sculptural qualities, all criteria that are fairly subjective in nature. First, an individual was chosen, views of which are show in figure 3.14 below.

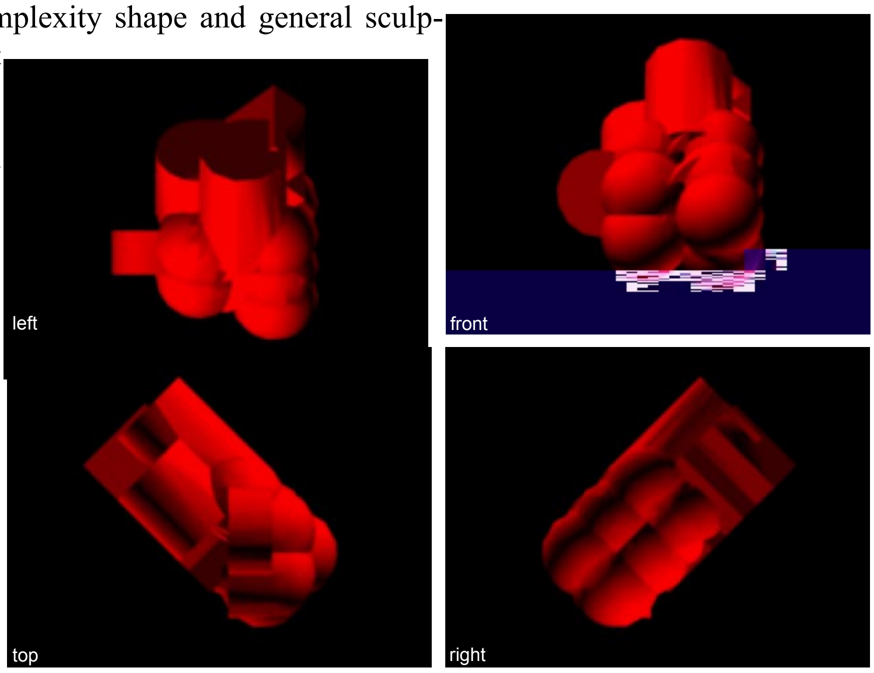
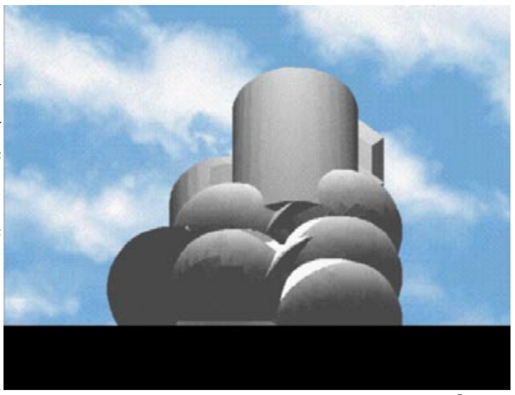


Figure 3.14 Selected individual

The individual was then hollowed out by placing a smaller copy of itself inside itself and performing the Autocad Boolean *subtract* operation on the two solids, creating a hollow shell solid. Figure 3.15 shows the solid with the smaller copy beside and inside it. The next step was to slice the solid to create a flat bottom so that the shape could sit on the ground as shown in figure 3.16



.Figure 3.17 shows the external views achieved by performing a quick rendering operation on the shape which has been placed on a ground plane. The object has now been re-

3.10 Exploring the Internal Space of the Buildoid

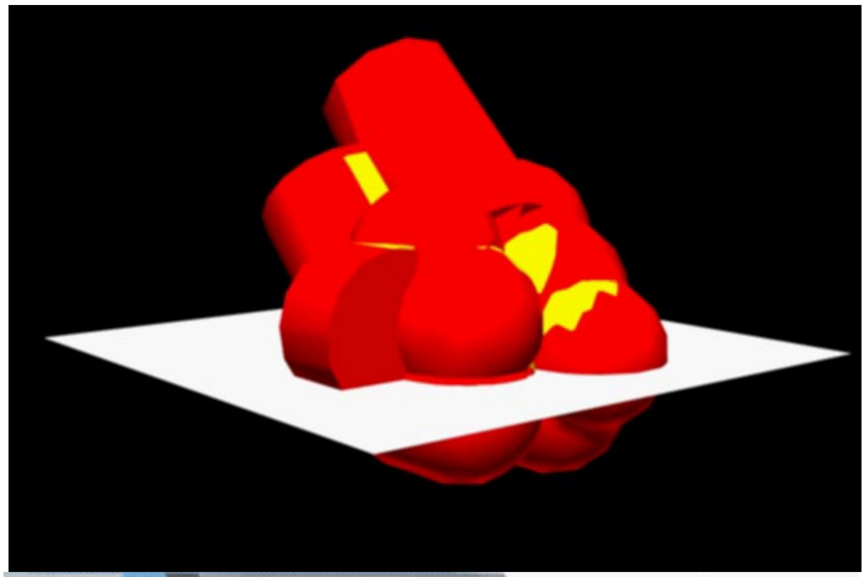


Figure 3.16 Slicing the individual to create a base to place on the ground plane.

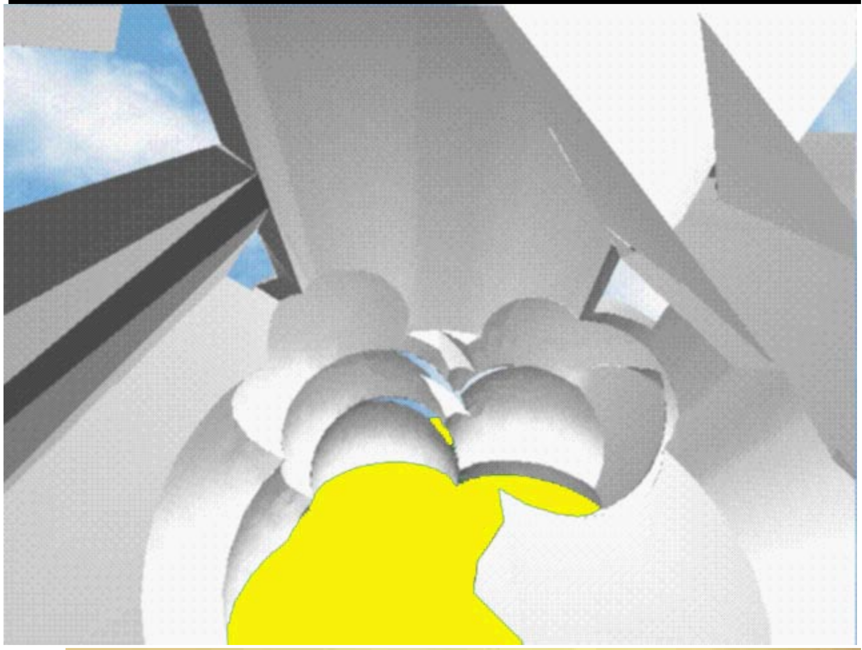
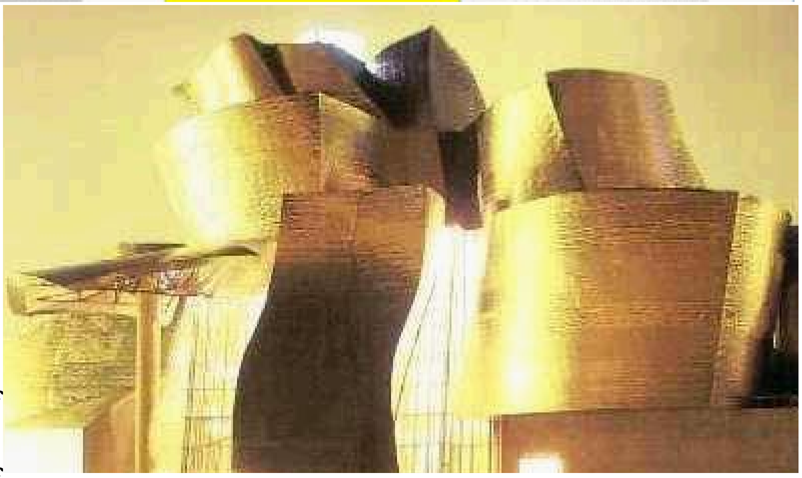


Figure 3.18 Interior view inside the buildoid looking down.



codified in an extremely simplistic and naive way to read as an architectural object not quite a building... more like a "*Buildoid*".

Figures3.18 and 3.19 above show two interior views of the Buildoid. The spaces are complex and reminiscent of the spaces created by a number of contemporary architects who create architecture as a kind of urban sculpture. One of the main exponents of such methods is the Californian Architect Frank Gehry whose Guggenheim Museum in Bilbao is perhaps his best known work. See figure

Figure 3.20

3.20.

4 The Surface Breeding Experiment

4.1 Breeding with Composite Surface Objects

The following experiment is similar to the Solid Breeding routine however this time the breeding is done with AutoCad surface objects. These are not standard primitives but are derived from standard profiles from which the surfaces are revolved.

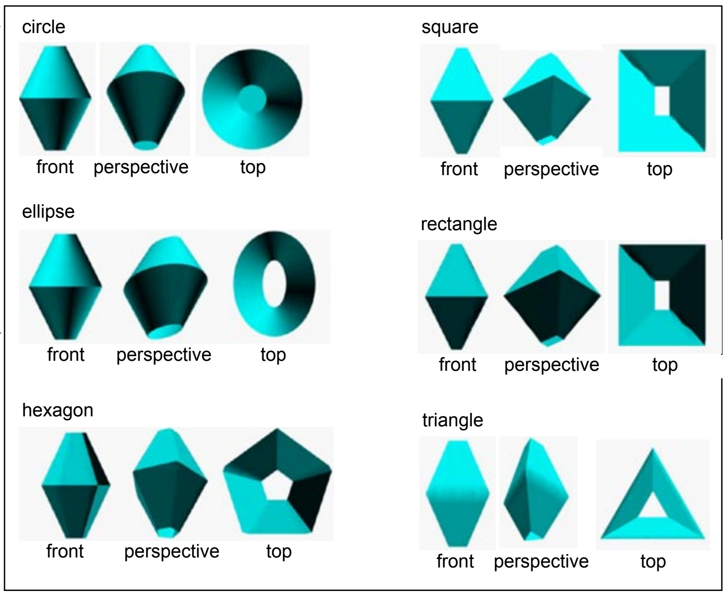
4.2 Creating a Composite Surface Object

Figure 4.0 shows how the surface object is composed of 10 rings, placed one on top of another. 11 profiles are created between which a revsurf operation is carried out to create the 10 rings. Figure 4.1 shows the six different types of composite objects. The objects are derived from the following standard two-dimensional profiles.

circle
ellipse
hexagon (5-sided polygon)
square
rectangle
triangle (3-sided polygon)

Right
Figure 4.0
Composite Surface Object.

Far right
Figure 4.1
Six different types of Surface Object



4.3 Function Set

The function set is smaller than the one used with the solids. This is because, the Boolean operations union, intersect, and subtract cannot be performed on surfaces, which are not solid objects.

move
copy
rotate
scale

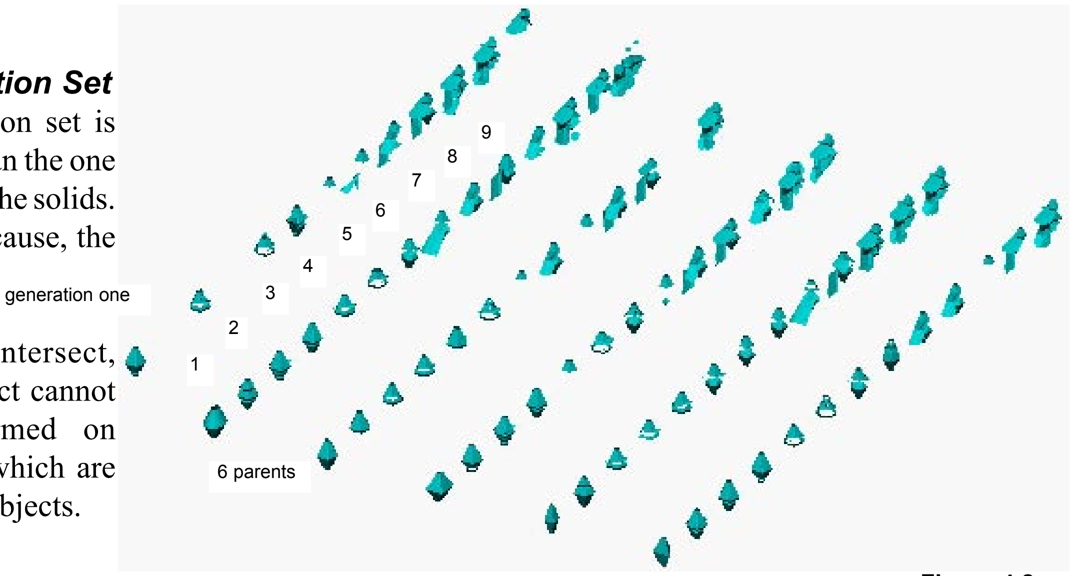
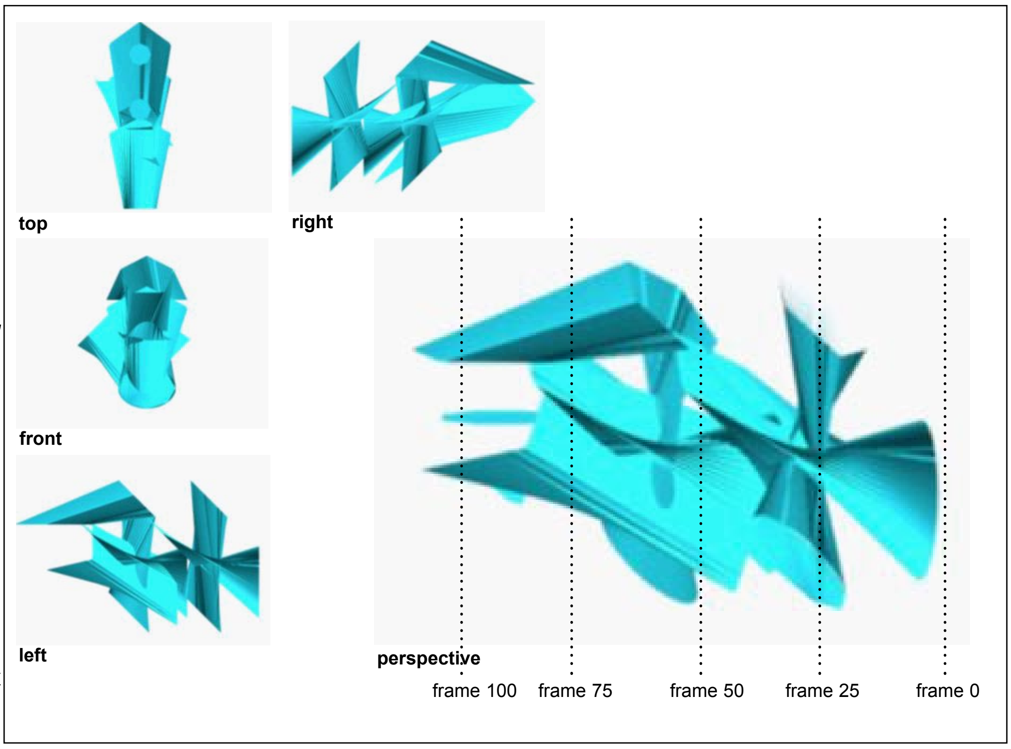


Figure 4.2

mirror

revsurf



4.4 Running the program

Figure 4.2

Figure 4.3 Several views of an evolved individual.

shows the result of running the surface breed program for thirteen generations. The six parents are in the bottom left corner.

4.5 Choosing an Individual

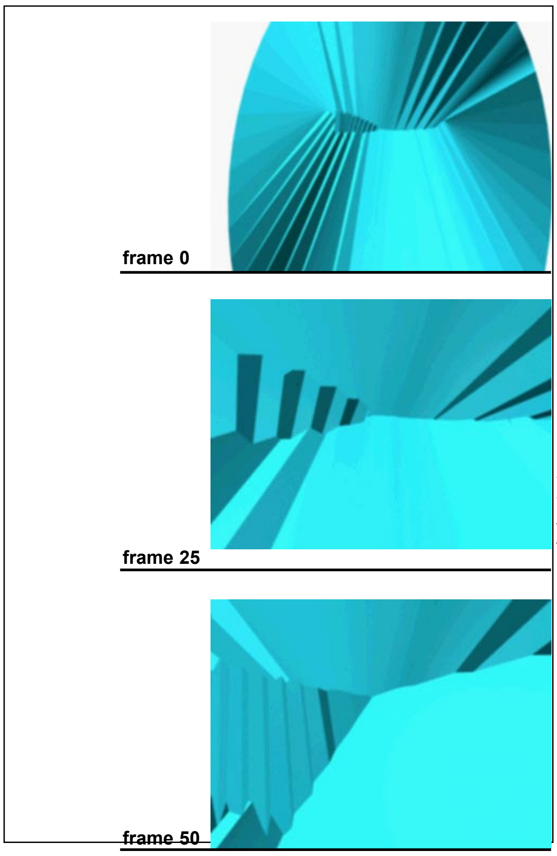


Figure 4.3 shows an individual which has been picked from the thirteenth generation. Unlike the solid object, the space enclosed by the surface is not continuous, but is infact a collection of spaces. In order to get a glimpse of the kind of spaces created inside the surfaces, five views were taken moving from right to left through the object. Figure 4.4 shows the five views.

Figure 4.4 Views moving through the individual

4.6 Views through the Surface Object

The surface breeding program creates a different type of object that lends itself less readily to re-coding as a buildoid. However it would not be difficult to take all or a portion of the object, scale it up and explore the architectural potential of the interior space and exterior form.

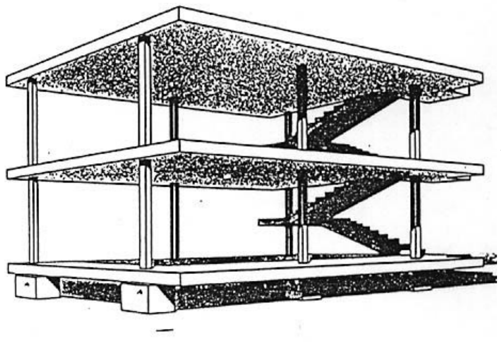


Figure 5.0 Domino House

5 Grammars of the dom-ino house:

5.1 Using GP of morphological functions to explore the space of grammars of form.

In the summer of 1998 the Baunetz architecture internet prize http://www.baunetz.de/internet-preis/ announced a student competition based around a reinterpretation / deconstruction of Le Corbusiers ‘Dom-Ino◊ house. One of the prize winners ("genetic bastards" Christoph Körner, Lars Krückeberg, Wolfram Put 1998[7]) was a lively essay on the (as yet unrealised) possibilities of recombining the elements of the standard house. We decided that if it was possible to define the domino house as an s-expression in the terms of the GP system we had already developed, then the evolution of the phenotype would offer just such instructive insights into the overall architectural possibilities inherent in the canonical morphology offered by Le Corbusier as the starting point for building.
The domino house was originally conceived by Le Corbusier as a concrete frame structure with vertical columns and an active reinforced slab (no beams as such).

In these experiments, the initial object was a simplified version of the canonical drawing, expressed as the result of making different shaped blocks and copy and move functions to get them into the right position.

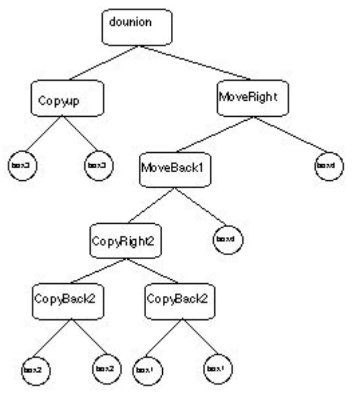


Figure 5.1 Additive Function Tree

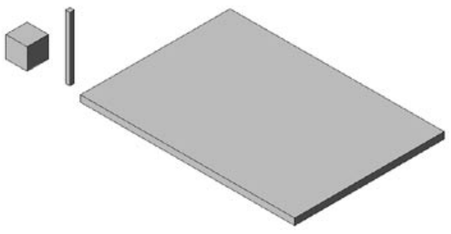


Fig. 5.2 Domino Parts

5.2 Experiment 1

Domino by block move & copy - the additive model

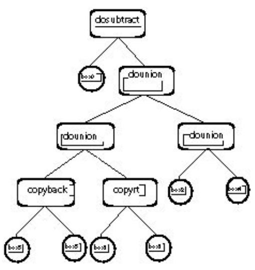


Figure 5.3
Subtractive Function Tree

The additive model constructs the basic building by using a vocabulary of slab and stick morphologies (fig 5.2) to form the basic building blocks. The functions used are shown in fig 5.1 and consist of moves and copies.

The use of dangling box4's at the end of the diagram is to satisfy the requirement that each morphological function needs two arguments, box 4 is a tiny "dummy" object to plug into the expression to mend this gap. Each phrase of this expression (copy up, move back etc) can be seen as a little "constructor" which corresponds to the transformations in a shape grammar. The blocks used were defined as 3 objects plus the extra box4. This small branching tree structure becomes the seed genotype for the evolution process to work on.

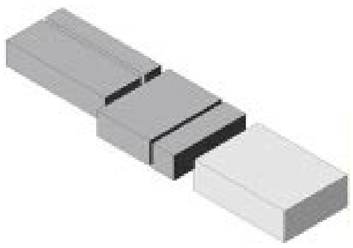
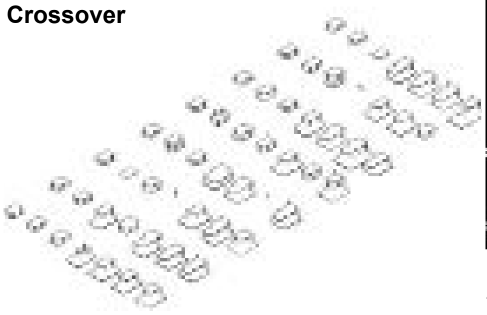


Figure 5.4
Domino Subtractive Parts

5.3 Experiment 2

Figure 5.6
Domino
Additive
Crossover



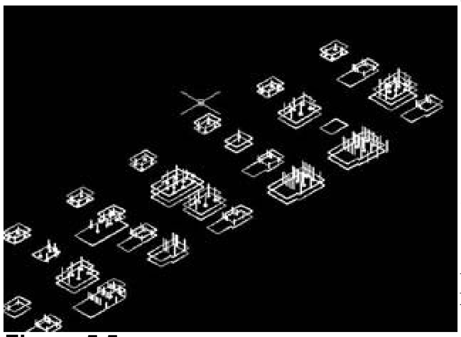


Figure 5.5
Artificial Breeding Using Additive Set

The domino house can also be constructed by using boolean operations. In this case the blocks are designed to carve out the domino morphology from one big block by boolean subtraction. The english version of the above expression is : "subtract the union of on the one hand the union of copyingback box3 & box3 and copyingright box1 and box1, and on the other hand the union of box2 and box4, from box 0."

There are five blocks , block 0 the one from which the object is carved (shaded pale grey in figure 5.4), and blocks 1 to 4 (+ plus copies) which are copied, unioned and subtracted from block 0.solid geometry) operations.

5.4 Experiments with the Additive Model

The GP system was set up with a lexicon consisting of :

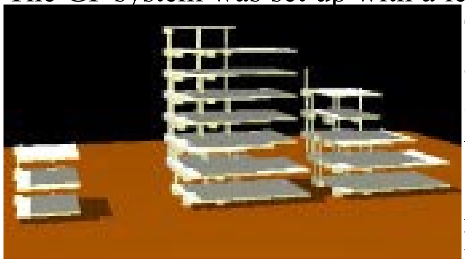


Figure 5.7
Additive Crossover and Mutate

The Function Set of 13 copy and move functions made up of the 5 in the original expression plus their symmetrical equivalents, and the 3 Boolean shape operators, one of which was used in the canonical model. The use of extra functions was to provide the GP system to explore a wider range of designs using the Mutate operation.
For the initial experiments only crossover was used, thus restricting the evolution/breeding to shuffling the canonical function set. Figure 6 shows the first four generations

using crossover only on two individuals per generation. The initial population are set manually to the domino expression. the second generation (next row towards the viewer) is the result of the evaluation of a further 8 expressions consisting of randomly shuffled subtrees of the domino gene. In this first generation , since all individuals in population 1 are identical genotypically, it makes no difference which two individuals are chosen. In later generations selection is made by eye on the basis of subjective judgements about the appearance of the individuals. Using only crossover for 7 generations (figure 5.6) of the additive model, and selecting for overall height and "sensibleness" by eye, the successive populations slowly increase in

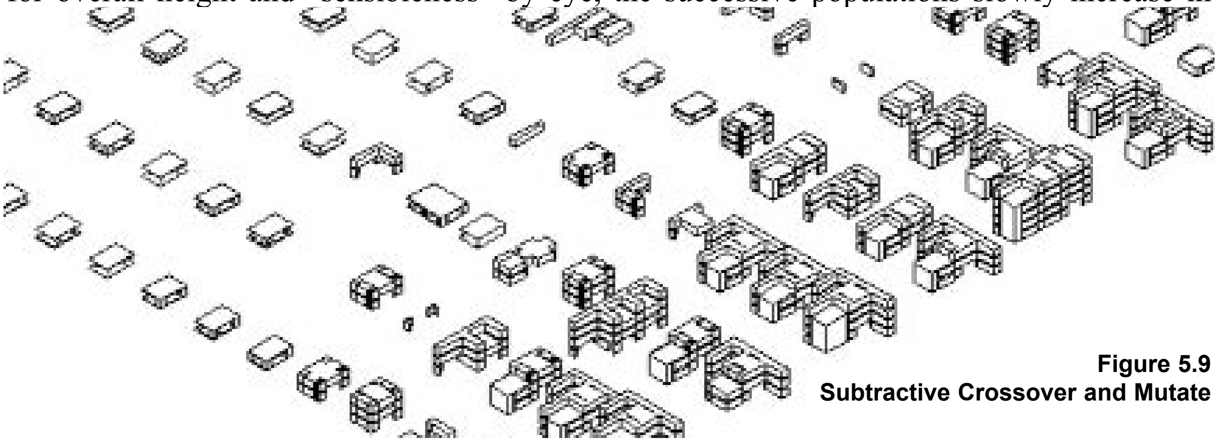


Figure 5.9
Subtractive Crossover and Mutate

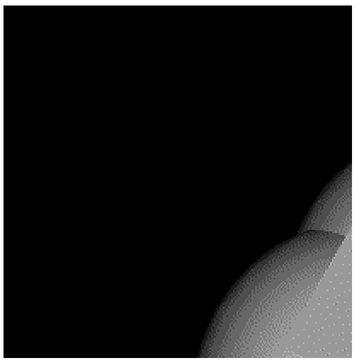
variety, but because the lexicon is restricted to the original set, the range of outcomes is often limited. Using a judicious mixture of crossover (to explore a closely related set of promising outcomes) and mutate (to force the inclusion of functions from the wider lexicon)it is possible to drive the system towards more extreme outcomes. Figure 5.7 shows members of the last (12th) generation where selection had been again for height and usefulness.

5.6 Crossover only with the subtractive model.

In this method of creating the domino geometry, the base set of functions is very small, and as a consequence the design space is restricted to single height arrangements. Where the cutting blocks move beyond the reach of the original positive◊ block 0 they survive and form agglomerations of large volumes. With Mutation also used a wider range of options is available. The illustration (figure 5.9) examples of 12 generations showing the evolution of multi storey units with structural bays.

6 Automatically Defined Functions

6.1 Saving the Subtrees



At the same time as conducting these experiments we also implemented the Genetic Library (Angeline,1994 [8]) idea, where small subtrees of successful parents are compressed and saved as a library of new functions. For example running the solid breeding routine using natural selection for 20 generations generated 64 ADF's the 10th one of which is defined as:

(ROTFT (TORUS1) (MOVEUP1 P1 P2))

Where P1 and P2 represent parameters to be filled out when the ADF is inserted into the genome. Figure 6.0 shows the phenotype for this ADF with the parameters replaced by the following components.

(FUNC11 (SPHERE1) (BOX1))

The likelihood of identical subtrees being embedded at different nodes in the same genotype (function tree) leads to fractal like self similarity at different levels of unfolding. This recursive embedding of morphological motifs at different scales or with different elements is a common property of architectural objects, responding to similarly nested social/ structural organisation of the brief.

It is a frequent criticism (see Bentley 99 for a discussion[9]) of GP that the crossover operation is too destructive to be useful, and that frequently good individuals are lost at early generations because the chunking is too small, based as it is on a tree of primitive functions.

Using Automatically Defined Functions allows the function set to be expanded during a run with multi parameter chunks that are seen as single nodes by the crossover and mutation operations. This has the effect of generating new "words" in the language (or more properly compound nouns or clauses).

The current implementation generates the AFDs automatically, selecting subtrees randomly from breeding pairs of parents, and adding their definitions to the function set, so that they can be introduced during mutation, which is usually set at some quite high level (10% –40 %). The new functions are multi parameter and are passed on by crossover. Angeline (94,96[8][10]) has shown that such ADFs can be observed to accumulate in genetic material producing successful phenotypes.

Early experiments show that the natural selection experiments show quite useful and continuous improvements in simple fitness measures, and that the amount of ADF material does appear to ameliorate the destructive effects of crossover.

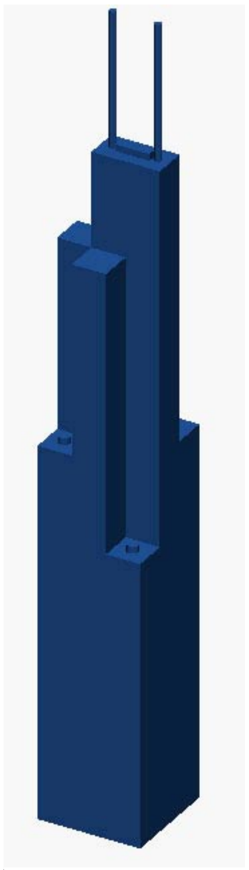
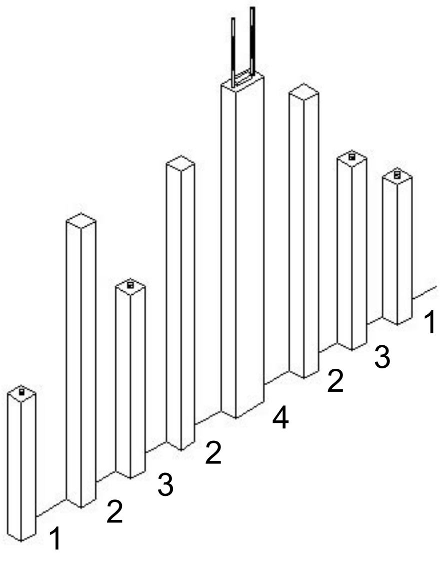


Figure 7.0
3D Model of Sears Tower by Skidmore Owings & Merrill Chicago 1974.



7 Interbreeding Buildings

7.1 Developing a Design Grammar

Left
Figure 7.1
The model broken down into 8 towers

The Sears Grammar

An initial experiment was conducted whereby a simplified three dimensional model of an existing building was taken and developing a grammar which would re-create that model using the same genetic routine as used with the Domino experiment. Sears Tower was chosen as an initial subject.

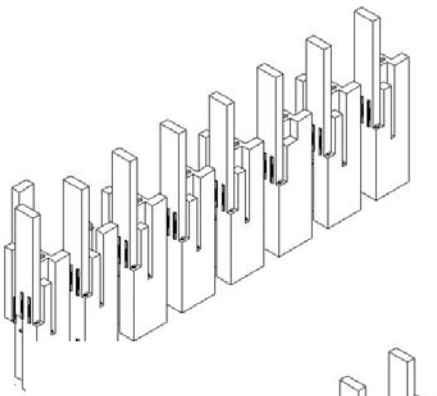
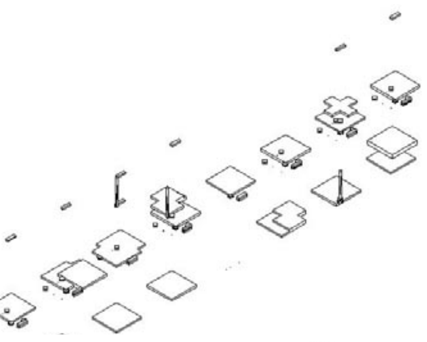
Generation method 1:

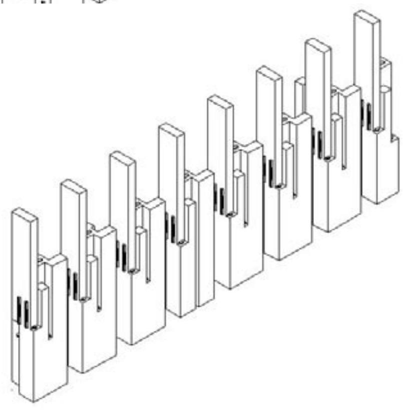
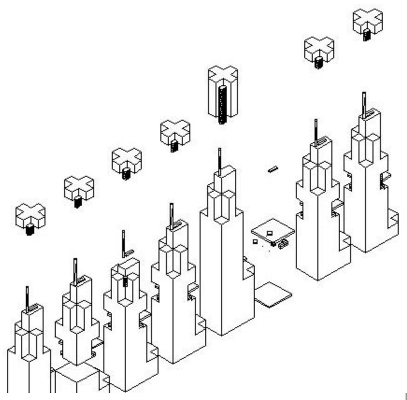
The first attempt to derive a grammar from the model involved breaking the model down into it’s constituent parts. The model was broken down into six smaller towers, a number of water tank volumes, and two chimneys.

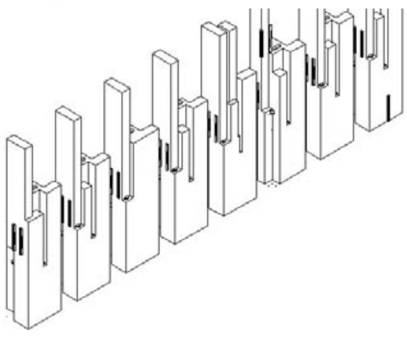
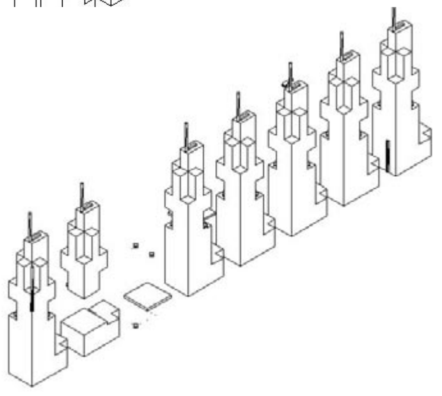
The tower was conceptualised as 8 separate towers, it was found that there were actually four different kinds of towers. Figure 7.0 shows the Sears model and figure 7.1 shows it the same model broken down into 8 separate towers of types 1 to 4. There are two occurrences of each type except type 4 of which there is only one.

7.3 The Sears Tower: Generation Method 2

The second method used to develop a grammar for the Tower was to conceptualise the building as a series of layers of differing profiles. The building was sliced into layers and it was found that the building is composed of four different sectional profiles throughout it's full height. This time four solids were created which were then stacked one on top of another to create the final tower.



Generation 8



Generation7



Generation 6

7.3 It's all in the coding...

Having established these two methods of creating a grammar for the same model, an experiment was carried out to illustrate how much effect the coding method has on the breeding results. Each coding was fed into the Breeding routine, to see what kind of progeny would be created. The following figure 7.5 shows what kind of individuals were created using the version one Sears Grammar (the tower method), on the right hand side and the version two grammar on the left hand side.

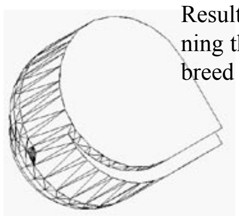
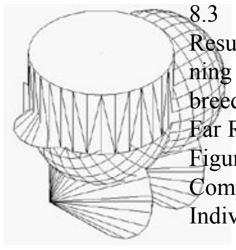
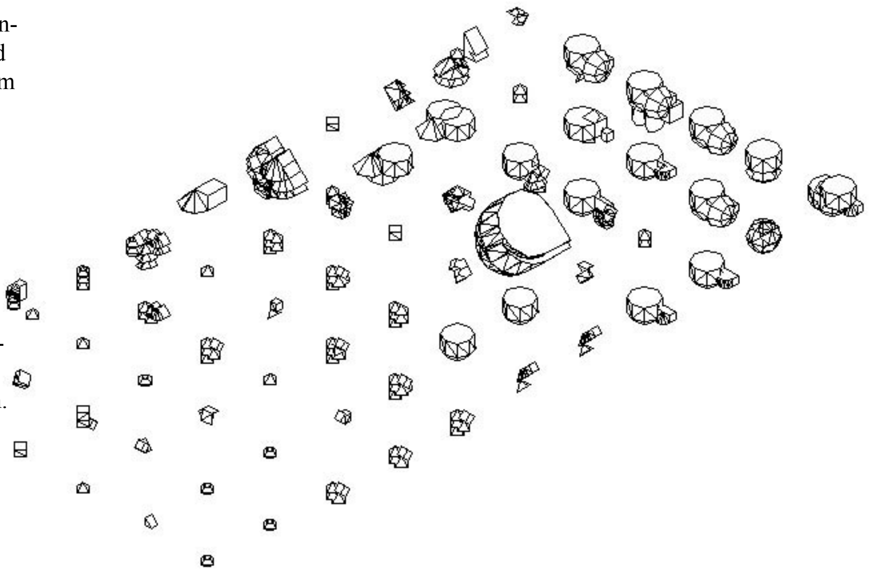


Figure 8.2
Results of running the solid breed program



Right, Figure 8.3
Results of running the solid breed program.
Far Right, Figure 8.4
Complex Individual.



individual B
<u>total surface area = 200</u>
no. of elements = 1
result = **200**

8 Survival of the Fittest

8.1 A proposal for a fitness function for the Solids Breeding Program.

A series of simple measures were tested to try to quantify the eyeball test for “interestingness”. The total surface area command , and a further refinement average surface area per object proved to at least to drive the system towards some complexity.

8.6 Conceptualising the Fitness Function

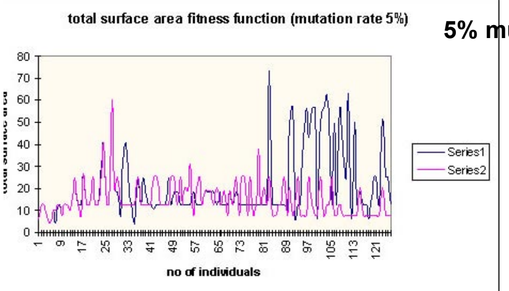
The surface area fitness function is an objective fitness function based on actual measurement. However it may be useful to conceptualise the function in different terms. The more surface area an individual has the more complex it seems to be. Figure 8.2 shows the results of running the solid breed program for 10 generations.

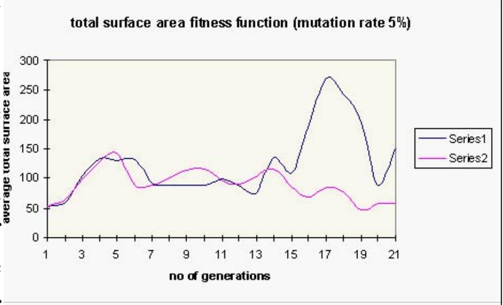
Looking at the results of the experiment, it could be argued that as generations increase the objects get more complex. Certainly the more complex the objects the larger the surface area, but there are exceptions when for example the scale function occurs and the result is an individual with a relatively large surface area, but fairly simple form. Figure 8.3 shows such a close up of such an individual.Generally however it can be seen that as generations increase individuals do get more complex. That is to say they are the result of a large number of unions of separate solids. Figure 8.4 shows a close up of such an individual from generation 10.

8.7 Complexity: Taking into account the number of elements in an individual

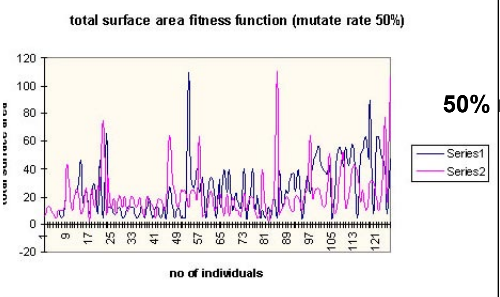
To take into account the number of individuals that the individual is made of we can modify the fitness function slightly to take the total surface area of the individual and divide it by the number of elements it is composed of. Consider figure 8.5 below, which shows an individual. It is infact composed of four separate elements. This individual will have a lower numerical score than a similar sized individual which is only composed of one element. For example take two hypo-

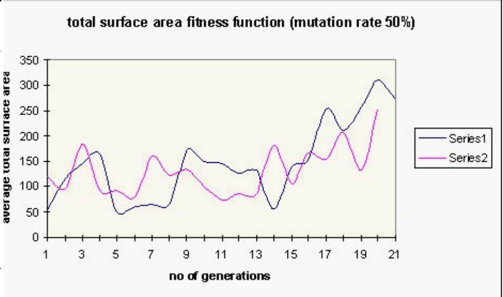
5% mutation rate



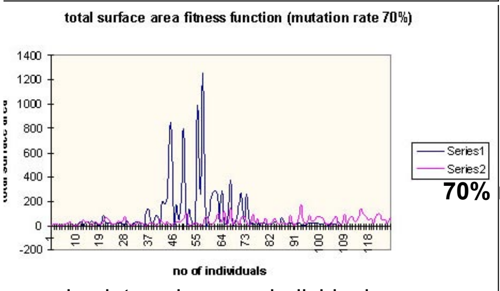


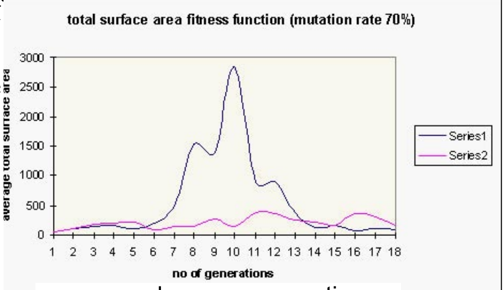
50% mutation rate





70% mutation rate





absolute values per individual

average values per generation

thetical individuals A and B

individual A

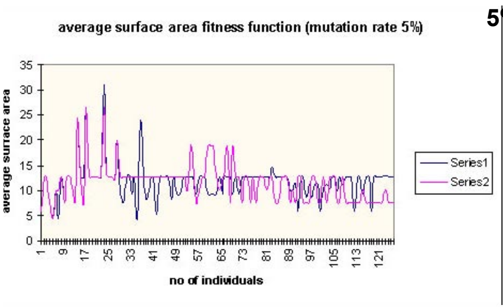
<u>total surface area = 200</u>

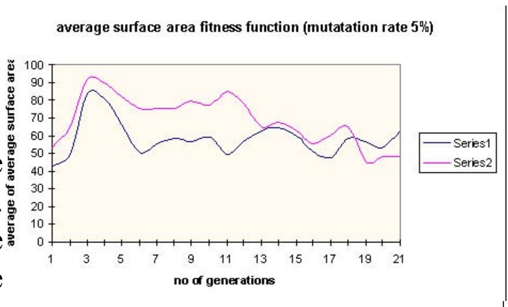
no. of elements = 4

result = **50**

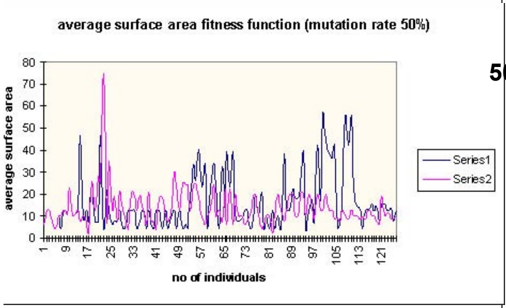
However we want to reward individual with *more* elements not less. As it stands individual B has the higher value. However we can alter the program to reverse the order of scores so that lower scores are selected rather than higher ones. This will mean that the *lower* the score (individual A) the *higher the fitness value.*

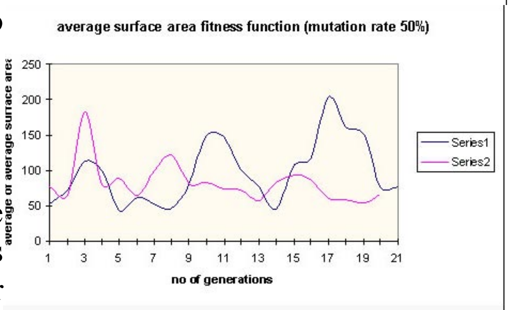
5% mutation rate



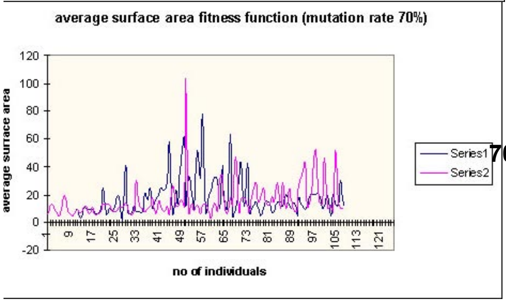


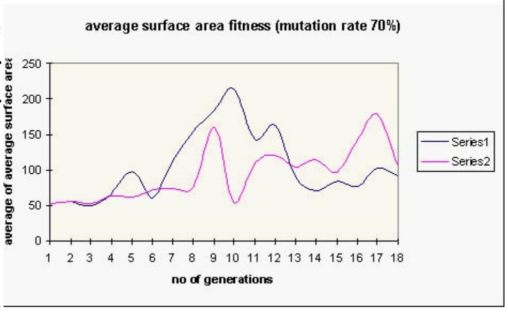
50% mutation rate





70% mutation rate





absolute values per individual

average values per generation

8.8 The Roulette Wheel Selection Method

The selection of individuals used with these fitness tests is known as fitness-proportionate selection, in which the expected number of times an individual is selected to produce is that individual's fitness divided by the average fitness of the population[11]. The roulette wheel method is used to implement this selection. Each individual is assigned a slice of a circular roulette wheel the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation.

8.9 Results of the experiments.

Two series of experiments were run with an initial population of 6 solid primitive individuals.

Series 1 -	using the total surface area fitness function
Series 2 -	using the average surface area fitness function

Two columns of data are shown for each series. The first column shows the absolute value for each individual. The second column shows the average value per generation. Both series are shown on each graph for comparison purposes and the solid breeding routine was run for 20 generations using a mutation rate of:

low - 5%
medium - 50%
high - 70%

Looking at the above results, we can see that the rate of mutation affects the fitness performance quite dramatically. A 5% mutation rate produces a marked peak in the series 1 data (dark blue). A 50% mutation rate produces a smoother increase and a 75% mutation rate produces a peak followed by a decrease in value. It would seem that the 50% rate is just enough to ensure that fresh combinations are created without losing valuable individuals that have a high level of fitness.

8.11 Results of the experiments:Series 2 Fitness Function: Average Surface Area

The series 2 experiment is looking to minimise the average surface area, hence the data curve is downward sloping. As with the series 1 experiment the 50% mutation rate yields the best results in that a consistent decrease in the average surface area can be observed.

9 Conclusion

The employment of a fitness function within a design genetic program is a difficult one due to the fact that design criteria are notoriously hard to quantify, particularly within an Autocad environment. Due to the limited amount of feedback available from a solid model, much of the selection criteria has remained on an aesthetic level or to do with simple parameters such as size and area.

David C Brown[12] has written lucidly on the strengths and weaknesses of the fitness function with Genetic Algorithms. He begins by endorsing the role that GA's have to play in doing configuration designs as carried out by Bentley and Wakefield, generating grammars that can generate good designs, and mixing aspects of existing designs to produce "creative" solutions as illustrated by the solid and surface series of experiments, and the hybrid series of experiments described in this paper He asks where the design knowledge resides in the GA - design equation. He goes onto point out that some of the knowledge is encoded in the population, some in the constraints used if any, but the bulk of the knowledge in terms of design is encoded in the method used to evaluate fitness. Brown states that having a single place to locate design knowledge is strength of the system, however the weakness of such a system is that "building an effective fitness function is very hard". This is because evaluating a design depends on many factors. All these factors have to be traded off against each other and furthermore constraints also have to be satisfied. Physical law, professional standards and codes, cost and strength are example of such constraints. To include all these aspects, and to generate complete designs would require a sophisticated knowledge based evaluation system embedded in the fitness function. If all this complex knowledge is compressed to a fitness value, Brown argues that it is becomes wasted knowledge. One possible way to overcome this would be to develop a hybrid between a knowledge based design system and a GA.

The hybridisation series of experiments carried out in section seven show how two buildings can give rise to a new order of designs which display the characteristics of both parents. Although these experiments are fairly crude in that the buildings are abstracted and codified to a certain level (3-dimensional building blocks), it is only a matter of computing power which decides at what level a building is encoded. That is to say it is more economical in terms of