Genetic Weapons

Filipe Afonso
Faculdade de Arquitectura, Urbanismo, Geografia e Artes
Universidade Lusófona de Humanidades e Tecnologias
376 Campo Grande
Lisboa 1749-024, PORTUGAL
E-mail: filipeafonso@arquilab-studio-ulht.com

Abstract

Architects, designers and artists have long dreamed of forms that behave like living organisms. From Frank Lloyd Wright's "organic architecture" to Archigram's "Living City" or Salvador Dali's visions of a soft and hairy architecture, they have imagined buildings designed to operate as natural entities. Design may be thus inspired by nature, evolving from a core of complex genetic mutations. What if buildings could be designed in the same way a cell develops into complex organisms from elemental forms? This suggests a direct comparison with the world of digital design, even the configuration of programming sequences, computational material or code able to generate and develop according to their own artificial computer "species". Genetic processes based on code are changing every aspect of design and construction, it being merely a matter of time before generative-self-assembly completely takes over. Real "genetic weapons" are now being used in the warfare of design. Evolution may reveal the way to a new conceptual design methodology taking as starting point computational processes, the expression of code oriented to geometry and form, producing in the end an antidote to preconceived architectures.

Introduction

The phrase 'genetic weapons' is no longer merely metaphorical, as today we are able to operate at the level of molecular action, affecting even the genetic blueprint, programming and designing natural living systems. By analogy with the design potential underlying biological processes, this paper attempts to explore new methods based on the manipulation of generative code for architectural production. Here the architect is seen as a process controller, acting from within the formative procedures of creating the architectural end-product. For this purpose the Sunwarm project was developed as an example contributing to the problematic identified above. It is an opportunity to investigate the relationship between architecture morphology and the generation of information structures using computational code to produce design dynamics. This project illustrates some of the key working methods available within the field of digital systems such as "generating" and "compositing" taking as starting point computational methods oriented to geometry, shape and form. The design goal of the project is to create a canopy system based on the aggregation of single structural elements, testing how evolutionary techniques of variation as expressions of code can be used to create complex simulated structures for architecture. A parallel with biological mechanisms is established, by correlating the "genetic" inputs, genotypes and programmed growth expressions based on code (applying Microsoft Visual Basic programming language), by learning how to manipulate coded matter and thereby progressing to architectural objects-geometries. The use of scripting techniques has proved rewarding since its ability to increase the number of outputs (solutions) by using simple instructions (grammatical rules) allowed the testing of a larger set of results.

Sunwarm "Natural" Breeding

Carrol [1] discusses the pathways of adaptive evolution and shows that genetic code variation among individuals is translated by the process of development into phenotypic variability. This process suggested that a similar logic could be applied to architectural/design production.
The paper considers the idea of generative impulses not only powerful enough to bring about the formation of biological entities as inspiration for architectural built structures, but also useful in simulations to create structures of higher levels of complexity that could be easily adapted for design purposes. To Holland [2] both biological and simulated digital evolutions involve the basic concepts of genotype and phenotype, and the processes of expression and selection. In biological systems the genotype is the genetic information that codes for the creation of an individual. In design simulated evolutions there are many possible representations of genotypes. For the purpose of this paper, design genotypes were defined as construction elements [Figure 1], that is to say, the very basic information necessary to simulate an architectural structure. In this way the, Sunwarm experiment relies on the performative capabilities of these single elements, or genotypes, as generative antecedents.

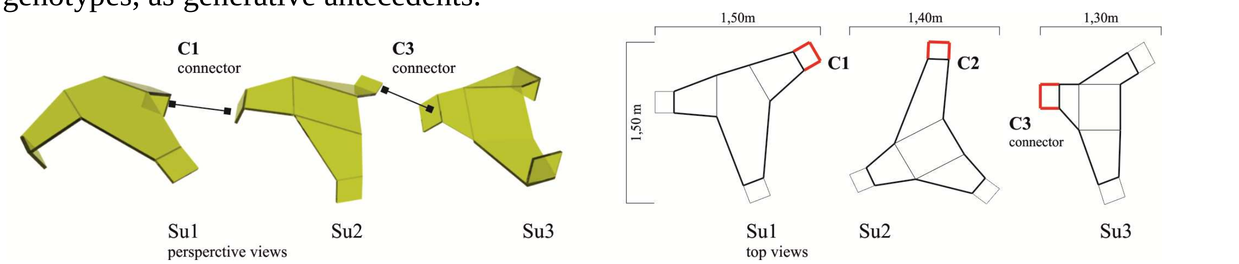

Figure 1: *Grammar of genotypes, three assembly units (Su1, Su2, Su3) for the production of the canopy system. These base elements may be produced with curved acrylic panels. Depending on the thickness of the acrylic material, the elements may present structural resistance contributing for the self supporting ability of the final aggregate.*

The phenotype is the individual itself, or the form that results from the developmental rules implicit in the genotype. Expression is the process by which the phenotype is generated from the genotype. Such a process, translated by procedural rules, relates together the three genotypical elements, aggregating them in incremental steps, using specific connective possibilities. Su1, Su2 and Su3 are juxtaposed face-to-face using the rectangular connector surface. Due to the individual shape of the elements and the positioning between them, the assembly system achieves three-dimensionality. The expression rules are specified in the VBScript [Figure 2] in order to allow selective connectivity by the user, designating the particular genotypes to be assembled. Using the script it is possible to drive the growth in specific directions, assigning base points and reference points of orientation in each element. This will imply asking the user to select those face-connectors that determine where the next connections will be made, in this way bringing about different assembly organizations [Figure 3]. These results were tested using three-dimensional representation software (Rhinoceros®) able to interpret and execute Microsoft Visual Basic programming language. Since it is difficult to analytically measure the aesthetic visual success of the simulated canopy systems, here the selection of the fittest phenotype is based on visual perception. Besides that, the aggregated structure must avoid the intersection of elements and prevent the system from drifting towards large numbers of units without necessarily improving the results as a shading mechanism, or increasing the difficulty of real material implementation in terms of physical structural behavior.

```
Call Sunwarm intel()

' definition of the main function to assign 3 points
in each element to further perform 3D alignments

Sub Sunwarm intel()

        Dim objArr(9), gens

        gens = Rhino.GetReal("numero iterações", 5)

        objArr(0) = Rhino.GetObject("objecto base")

        objArr(n+1) = Rhino.GetObject("Base ponto referencia 1", 1)
        objArr(n+2) = Rhino.GetObject("Base ponto referencia 2", 1)
        objArr(n+3) = Rhino.GetObject("Base ponto referencia 3", 1)


        objArr(n+4) = Rhino.GetObject("target 1 ponto referencia 1", 1)
        objArr(n+5) = Rhino.GetObject("target 1 ponto referencia 2", 1)
        objArr(n+6) = Rhino.GetObject("target 1 ponto referencia 3", 1)
        objArr(n+7) = Rhino.GetObject("target 2 ponto referencia 1", 1)
        objArr(n+8) = Rhino.GetObject("target 2 ponto referencia 2", 1)
        objArr(n+9) = Rhino.GetObject("target 2 ponto referencia 3", 1)
        sunwarm objArr, gens
End Sub
' definition of the function to orient three-dimensional elements
Function sunwarm(objArr, gens)

        Dim counter, i, baseArrPts(), tarArrPts1(), tarArrPts2(),

If gens > 0 Then
                ' location of base points
                counter = 0
                For i = 1 To 3
                        ReDim Preserve baseArrPts(counter)
                        baseArrPts(counter) = Rhino.PointCoordinates(objArr(i))
                        counter = counter +1
                Next

                ' location of target1 points
                counter = 0
                For i = 4 To 6
                        ReDim Preserve tarArrPts1(counter)
                        tarArrPts1(counter) = Rhino.PointCoordinates(objArr(i))
                        counter = counter +1
                Next

                ' location of target2 points
                counter = 0
                For i = 7 To 9
                        ReDim Preserve tarArrPts2(counter)
                        tarArrPts2(counter) = Rhino.PointCoordinates(objArr(i))
                Next

                'elements orientation
                objArr1 = Rhino.OrientObjects(objArr, baseArrPts, tarArrPts1, 1)
                objArr2 = Rhino.OrientObjects(objArr, baseArrPts, tarArrPts2, 1)
        End If
End Function
```

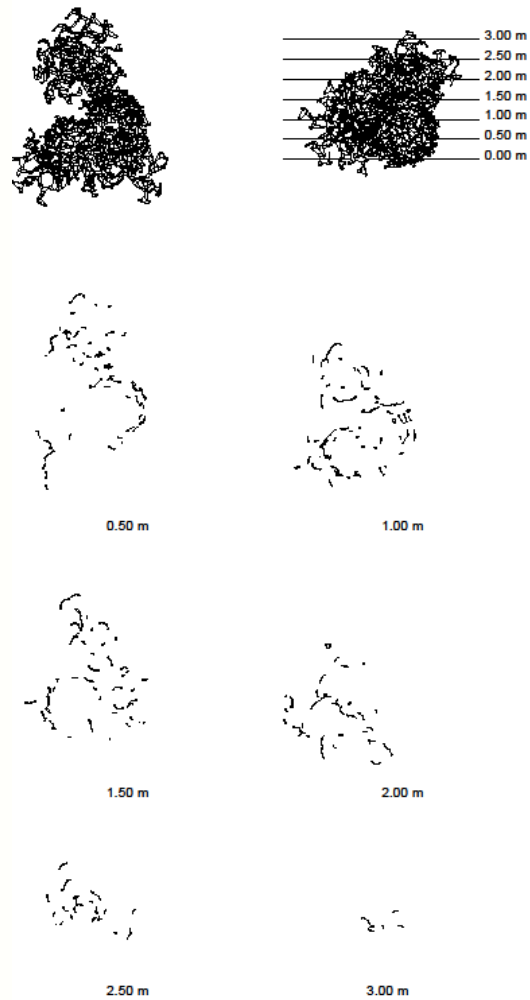

Figure 2: *Script translating the aggregation rules. The blue commentations explain the parts of the code executing different tasks in the three-dimensional orientation between elements Su1, Su2 and Su3. Example of phenotypical aggregate from the script. Top and side views. Sections at different levels.*

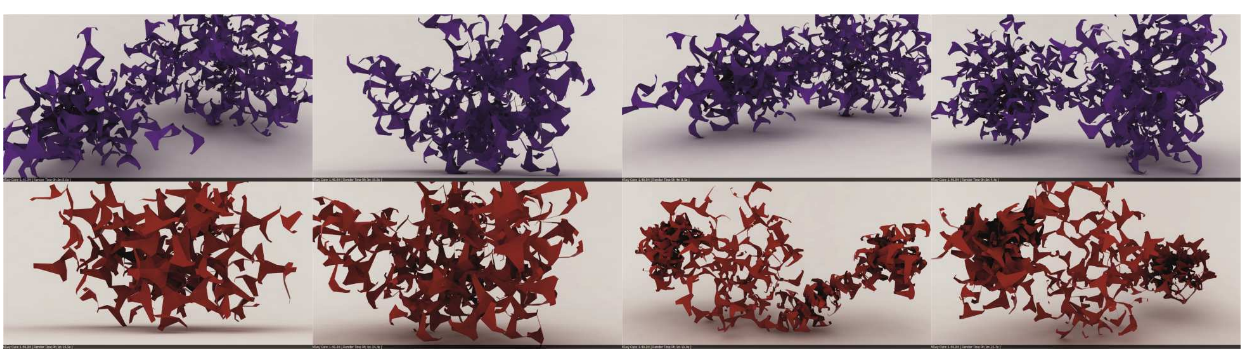

Figure 3: *Sequence of experiments applying the script written in Microsoft Visual Basic programming language. Contact rules and orientation between the elements Su1, Su2, Su3 are defined by the user as a first input to generate a desirable number of iterations of the script.*

Conclusion

The most recent digital and computational advances are subjecting design practices and logics to a fundamental revision. New possibilities are emerging from the redefining of working processes. New materials, new tools and new processes ought necessarily to lead to new architectures. In this way architecture is configured not as a static juxtaposition of independent layers, but as an open process, with specific rules and levels of strict interdependency. Lynn [3] points out the need of architects and designers to use a dynamic design methodology based on simulation systems and Kelly [4] argues that evolution is a structure of organized change, a mechanism capable of generating ever-diversifying and unexpected results. Perhaps the most surprising insight made possible by this project was that in every experiment conducted during the design process it proved impossible to anticipate the final configuration, even when using the same set of 'genes', the same script with defined aggregation and growth rules (production rules) [Figure 4].

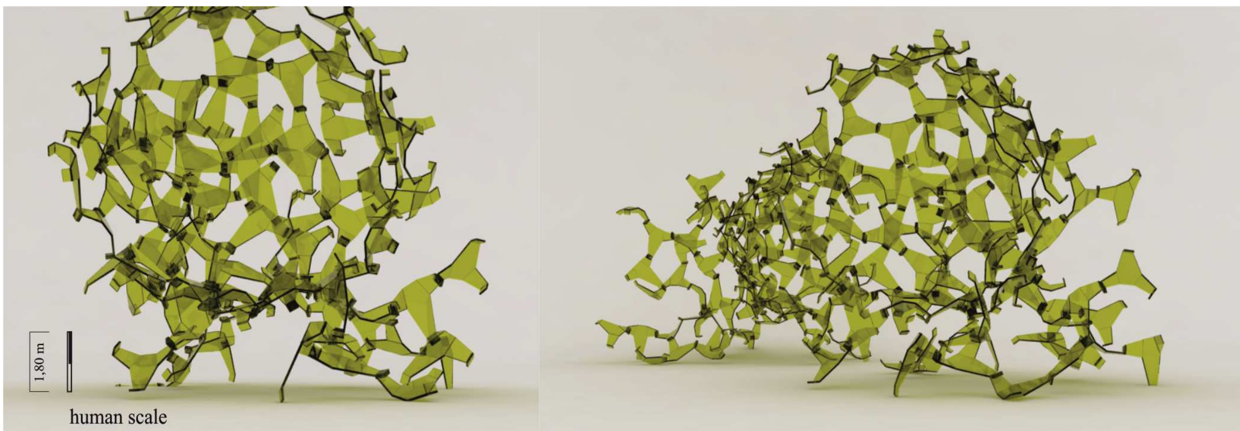

Figure 4: *Possible future, final aggregate canopy structure. Perspective views.*

Within the architectonic horizon of code, working processes may generate unpredictable results, outside of the designer's expectations. This is an alternative definition of architecture, where the designer is not only a person in possession of a brilliant idea but becomes a part of the process he initiates. The Sunwarm project defies the traditional view of architectural conception, redefining it as working towards a phenotype, a manifestation of form shaped according to the conditional logic of a process of production.

References

[1] Sean Carrol, *Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom*, Phoenix - Orion Books editions, London 2007.
[2] John Holland "Algorithms: computer programs that evolve in ways that resemble natural selection can solve complex problems even their creators do not fully understand", in *Scientific American*, 1992.
[3] Greg Lynn "An Advanced Form of Movement", in *Architecture after Geometry*, Architectural Design, Vol. 67 no. 5/6 May-June 1997, London.
[4] Kevin Kelly, *Out of Control: The New Biology of Machines*. Fourth State, London 1994.