Design by Computation

A. van der Zee
Department of Architecture, Building and planning, Design Systems,
Eindhoven University of Technology, Eindhoven, the Netherlands
e-mail: a.v.d.zee @tue.nl

B. de Vries
Department of Architecture, Building and planning, Design Systems,
Eindhoven University of Technology, Eindhoven, the Netherlands.

Abstract

In our paper we put Generative Design in a historical context. We will define Generative Design and briefly discuss the most important mathematical methods that have been developed, that are relevant with regard to shape generation. Often these methods don't have any relation to architecture, but are nevertheless very valuable when applying Generative Design.
In the second part of our paper will show Generative Design can be used in architectural training. A link is necessary between the traditional way of visual modeling and the mathematical formulae. To establish this link we use CAD scripting languages. The problem however is twofold: (1) scripting languages are meant to extend the system functionality and not to easily generate shapes and (2) most architectural students have no programming skills at all. We will show some the results of our "Generative Design" design atelier and most importantly explain the underlying mathematical approach that was encoded in the script.
Finally, we will summarize our experience with this studio and provide the guidelines for successful application of Generative Design in architectural design.

1. Introduction

The microprocessor, which was developed in the early eighties, is the root of the desktop computer. Because of the broad acceptance and low cost of desktop computers, the demand for software grew rapidly. For the architectural designing process this resulted in the development of graphical software, first mainly 2D but nowadays 3D.
At the end of the eighties research started to create new shape/forms which only could be drawn by computers, so called NURBS, non-uniform rational B-spline, (see figure 1.1) These 3D-shapes are the graphical representation of complex mathematical formulas. Today, a growing number of architects use these techniques, with well-known representatives such as Kas Oosterhuis, NOX and UN-studio.

Recently there is a renewed interest in the use of computer software during the form finding phase in the architectural design process, named “generative designing”.

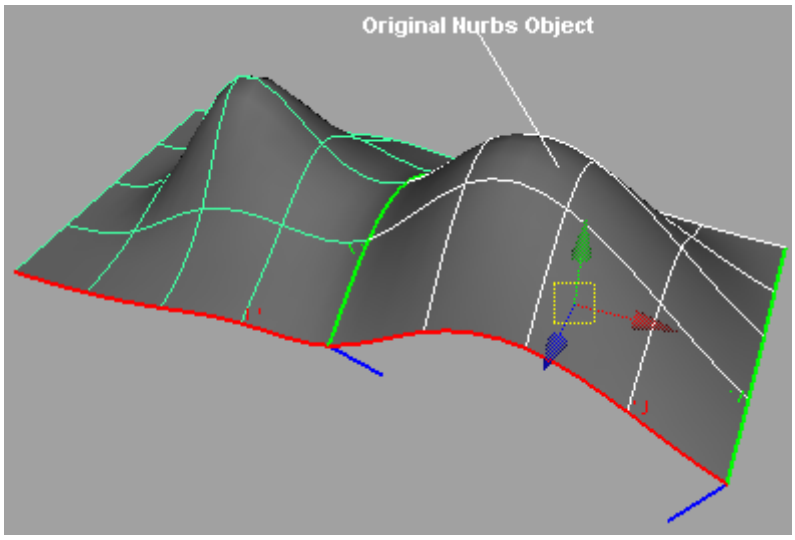


Figure 1.1 a simple NURBS surface

Terzidis (2008) refers to this renewed interest as the shift from computerization to computation. According to Terzidis computerization is the act of “entities or processes that are already conceptualized in the designer’s mind are entered, manipulated, or stored on a computer system.” In contrast, computation or computing, shapes aren’t manipulated by the designer but generated by the computer. Mouse-based operations on objects, as done in well-known CAD applications, such as AutoCAD, ArchiCAD etc, aren’t computations but belong to the ?act? of computerization. By using the embedded scripting languages of CAD-applications like VBA or AutoLisp in AutoCAD, the designers go beyond the limitations of the built-in functions. In a discussion with several leading scientist in the “CAD-field” (Kolarevic, 2005, pp 296) from the audience Ulrich Flemming formulated it as follows: “I would like to challenge the notion that you program only as a last resort in case software vendors don’t provide you with the right tools…. I wish architect would abandon this passive stance in which they simply accept what the software vendors offer them. They don’t even make suggestions as to how to improve the software; they don’t understand the software at the level at which you need to understand it if you want to make intelligent suggestions.”
So in the development of CAD-software the first range of meaning of this new technology was expressed not from “technical rationality” but from the past practices of users (A. Rahim in Kolarevic, 2005, pp 201). The current generation architect students never learned designing with pencil and ruler, so their expressing will be different to the former generation’s architects. The act of “computerization” will stay important in design practice, but there will be more and more demand for “computation”, as the new generation designers will go beyond the limitations given by standard CAD-software.
Because of this shift from computerization towards computation the following questions come to mind:

- What does computation mean in design or what is generative design;
- Are there ‘standard’ mathematical models which can be used?

- How can one use generative designing in architectural design? Or is designing not becoming a mere running of a piece of software.

In the next paragraphs we will address these questions.

2. Research

2.1 What is generative design?

Generative designing isn't a design process in the traditional way, but it is the use of a combination of different arithmetic methods in order to generate a set of different alternative solutions for the design problem at hand. In this way it becomes possible to find solutions to complex problems which in a traditional way of problem solving can't be found.

Galanter defined generative art as follows:

> *"Generative art refers to any practice where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set into motion with some degree of autonomy contributing to or resulting in a completed work of art."* (Galanter, 2003)

This definition is also applicable for architecture, in most cases it is a computer program (script) which is used instead of a set of natural language rules. These computer programs have the same underlying design pattern:

1) Describe the initial state;
2) This state will be modified according to a well describe set of rules;
3) Evaluation of constraints; these constraints test if alternatives fulfill the design goals;
4) This new state, derived in step 2, will become the initial state for the next iteration.

This step-by-step plan can be fully automated, in which case the designer decides when to stop the process. However, one or more steps can be accomplished by the designer self, in which case the designer becomes part of the iteration process.
The basic parts of the procedure mentioned above, are "state" and "rule-set". The state describes the architectural design; the rule set describes the transformations and denotes the constraints which the design has to fulfill in order to be an acceptable solution. This initial state will be, depending of its degree of freedom, transformed into alternatives of the initial design.
This formal approach resembles, not by coincidence, a big similarity with theoretical design methods as well practical design methods. These, most often complex, problems can't be solved by human's designers in a reasonable amount of time, if at all. However, generative designing makes use of the number crunching power and calculation speed of computers, to solve the design problems at hand.

2.2 Categories

Generative designing is based on a number of arithmetic models, as explained in the previous paragraph. These arithmetic models or better these algorithms can be

grouped in a few categories. These categories are only indicative, because most of these algorithms aren't use separately but mixed, which blur the classification boundaries. For simplicity we use the following categories:

- Parametric design (see 2.2.1)
- Cellular automata (see 2.2.2)
- Flocking of birds (see 2.2.3)
- Genetic algorithms (see 2.2.4)
- Shape grammar (see 2.2.5)

In the following sub-paragraphs we will discuss shortly the above mentioned categories and their underlying algorithms.

2.2.1 Parametric designing

In parametric designing, a geometric form (like a building component or building layout) is denoted as a set of depended variables or relations. By changing one or more of these variables or parameters, alternative forms are generated. Each of these generated forms can be checked upon functionality, usability etc. If a solution doesn't prove to fulfill all constrains, a new alternative solution can easily be generated and checked, by changing one or more parameters. This process continues till a satisfactory solution is found. The designer is in complete control of the process. The CA(A)D applications ArchiCad and Revit are parametric design tools. The designer can create different solutions to a design problem by manipulation different parameters in a relative easy way.

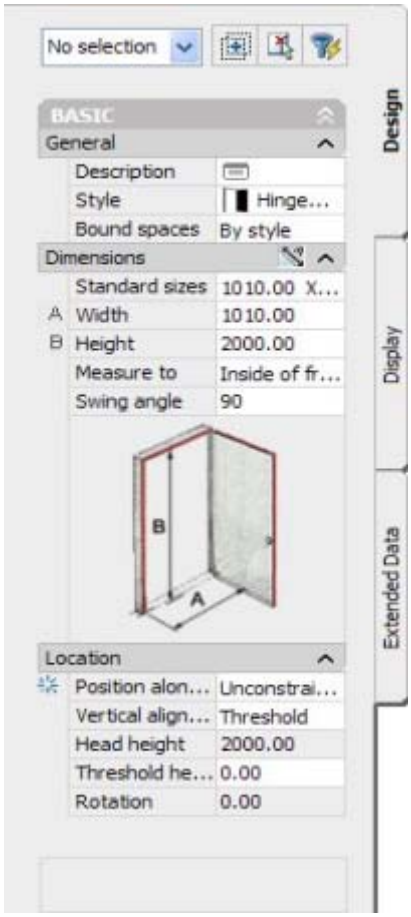


Figure 2.1 Parametric component

2.2.2 Cellular automata

Wolfram was for his age a genius; he proved that complexity can be derived from a set of simple rules and structures. Wolfram developed an application consisting of a grid in which every cell is ‘on’ or ‘off’ (dead or alive, 0 or 1). The application starts with an initial configuration, the state of the next row of cells is derived from the previous row of cells. There are 8 possible combinations for Row A defines the next state of row B in 000, 001, 010, 011, 100, 101, 110 and 111. The end result depends which rules are chosen, in this case there are 256 possible rule sets.

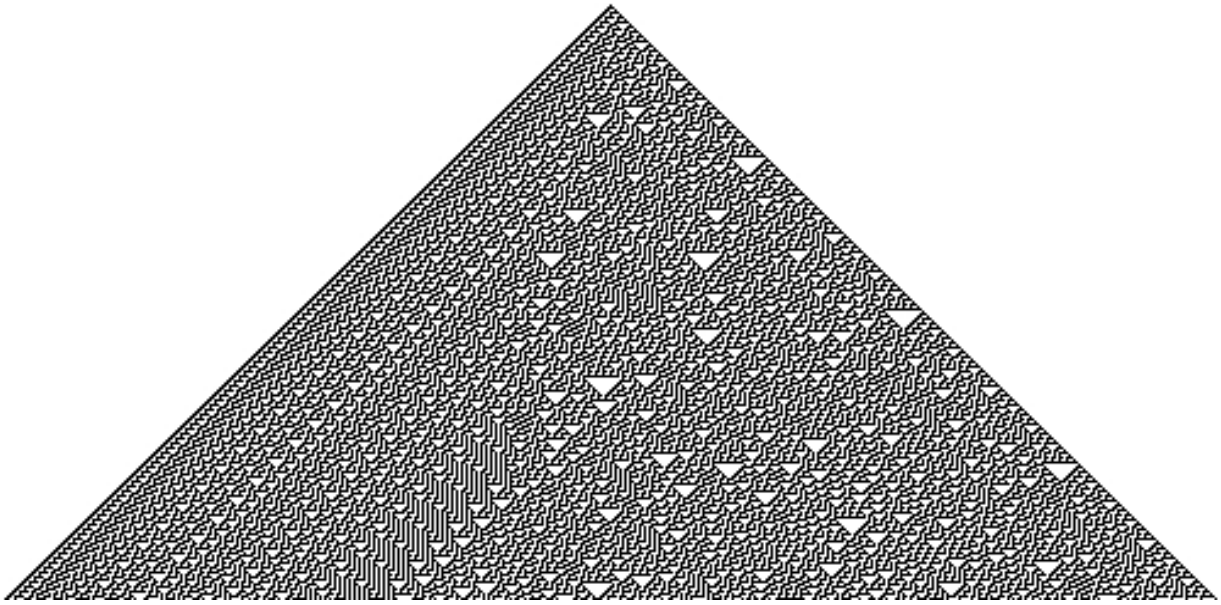


Figure 2.2 Cellular automata

A derived algorithm of the cellular automata algorithm is the so called 'Conway's game of life'. This algorithm is a classical application of an artificial life application (= A-life). Conway's game of life consists of a set of simple rules, implied on a grid of cells. A cell can't be alive if there are more then 3 or less then 2 adjective cells. A new cell will come to life when an empty cell is surrounded by 3 living cells. This simple rule set results in a complex behavior. The system will converge to a stable state, by dying of all cells or by reaching a steady state.

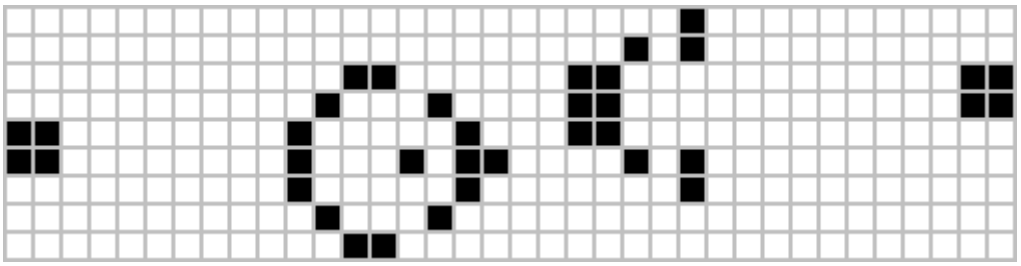


Figure 2.3 Game of Life

2.2.3 Flocking

A different area in which the A-life community is active is that of animal behavior, and the formulation of mathematical formulas which resemblance this behavior. A well know study is that of Graig Reynolds, named 'Boids' - an application that the flocking behavior of birds mimics in a realistic way. The algorithm is based on 3 very simple rules and results in an incredible realistic behavior. These 3 rules are:

- Separation: Steer to avoid crowding local flock mates.
- Alignment: Steer towards the average heading of local flock mates.

- Cohesion: Steer to move towards average position of local flock mates.

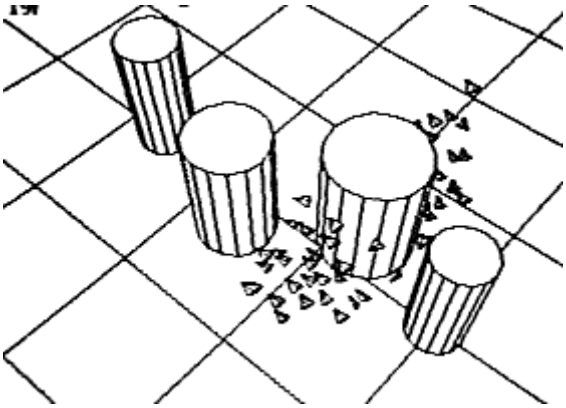
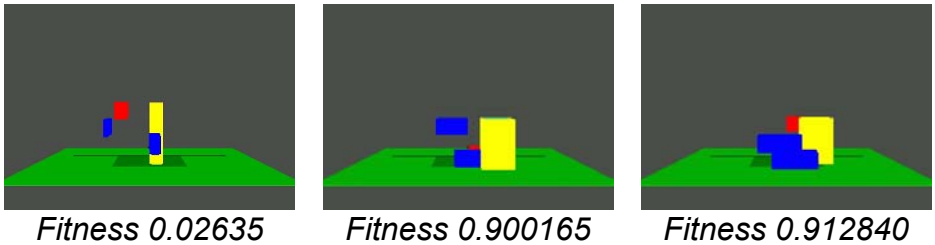


Figure 2.4 Flocking

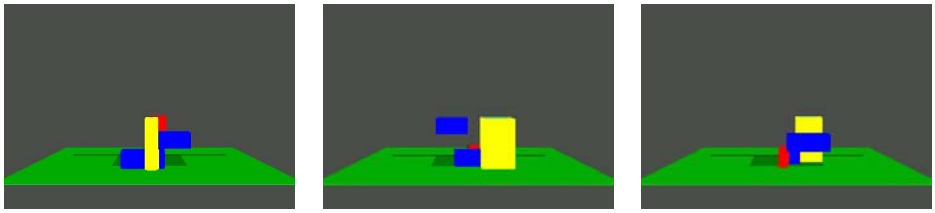
With use of these three rules the behavior of one bird is mimicked. If applied to a number of birds, the behavior of a flock of birds is simulated.

2.2.4 Genetic algorithm

In the sixties Holland (Holland, 1992) describes an algorithm based on genetic reproduction. A problem is parameterized, and these parameters are 'collected' in a string of numbers. By filling this string of numbers (= genotype), with randomly chosen numbers, one gets an indiscriminate solution (= phenotype) to the problem. Because of the process of iteration the string with numbers will improve and subsequently a better solution to the problem at hand will be found. The algorithm is capable of finding solutions for optimizations problems, especially for problems with different, sometimes conflicting constrains. For research on this topic see van der Zee en de Vries (2004)



Fitness 0.02635 *Fitness 0.900165* *Fitness 0.912840*



Fitness 0.95595 *Fitness 0.961112* *Fitness 0.961695*

Figure 2.5 Genetic algorithm

2.2.5 Shape grammars

A ‘shape grammar’ consists of a number of shape rules and one ‘generation engine’ which selects and processes the shape rules. A shape rule defines in which way an existing shape (or part of a shape) will be transformed. A shape rule is built up from two parts, separated by a right pointing arrow. The left side of the arrow is called ‘Left-Hand Side’ (=LHS). On the LHS are the conditions in terms of ‘shape’ and ‘marker’. The right hand side is called ‘Right-Hand Side’ (=RHS). On this side of the arrow is the transformation of the LHS written down, and where the new marker is positioned. The marker is use to localize and orientated the new form.

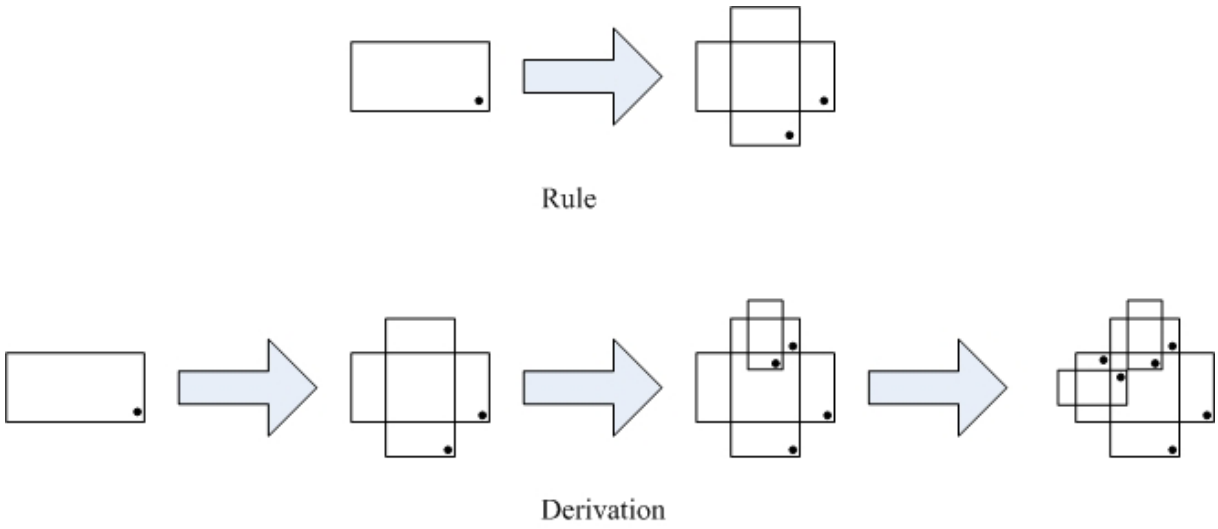


Figure 2.6 Shape Grammars

A shape grammar is build of at least 3 shape rules: a start rule, at least one transformation rule and a termination rule. The start rule is used to start the process of shape generation; the termination is used to stop this process. The simplest termination rule is a rule in which the marker is deleted.
Shape grammars are frequently used in historical architectural research on Palladian Villa’s and Victorian windows, as well in the design of new designs like the Malagueira Dwellings project by Alavaro Siza (Duarte, 2000). The shape grammar for a Palladian villa consists of 69 rules (Mitchell, 1990).

2.3 Implementation

As seen in the previous paragraphs all the methods start with an abstraction of the problem, this abstraction is the starting point of the process to come to a useable algorithm.
De building engineer who uses one of these models must have good understanding of the theory behind these models, in order to make a good abstraction. Specifically the representations of rooms as rectangular volumes, walls as lines, etc are not evident. Evaluation is only possible with these data in order to calculate distances

between rooms, outside area, volume of a room etc. Following this abstraction process, the algorithm can be implemented in CAD software by translating the model into ‘a script’ using an embedded script language. After a run of the script, alternative solutions will be visible on the screen. If the designer approved the generated solution, the process stops, otherwise the cycle is rerun. Finally the abstract solution must be refined into more detail for practical usage. So the outcome of the ‘software’ isn’t the outcome of the process, it is mere a starting point. The software helps the designer in finding a starting point of his/her own design process.

3. Education

Since the start of the academic year 2006 there we conduct, at the Eindhoven University of Technology, a design atelier ‘Generative Design’. This atelier is held in 3rd year of the bachelors. During this atelier the students do research on the feasibility of generative design.
In the next paragraph we will explore shortly the organization of the design studio. Then we continue with some examples of student work.

3.1 Design atelier organization

The design studio lasts 12 weeks, with a general meeting every week, at the first meeting there is an explanation about generative design and what it stands for. The first two weeks the students perform a literature research on the topic, to trace their on field of interest. During this period the students phrase their own design problem, which has to be solved with use of scripting. Together with the supervisors, CA(A)D software and the initial algorithm are chosen. Because of the familiarity of most students with AutoCad, this application is mostly use together with the VBA scripting language which is implemented in AutoCad, but also Generative Components or Rhinoceros are used.
The design problem is reviewed, because it has to be a problem in which ?generative designing? can be fully explored. Students are introduced to scripting with help of simple exercises. These exercises are also a start of the design problem they want to script. For simplicity the initial exercise is chosen in 2D. During the course, the results and progress are weekly monitored, and students get support with their scripting tasks. At the end of the design studio, the digital model will be used to create a physical prototype, with help of a laser cutter, milling machine or a 3D printer.

3.2 Student work

To clarify the meaning of generative design, we discuss the projects of a few students. As mentioned in previous paragraphs, the outcome of the design process is generated by a script made in AutoCAD. We don’t discuss the script itself, but describe the theoretical principles behind the script and the design. The text is taken from the student’s works.

3.2.1 Dom. Van der Laan (M. Burgman)

“If you get a design assignment, most of the times the function is the main focus point of the design- process. What to do if one is given the opportunity to choose the function of the building itself, and the only criteria is that the design has to be scripted, in order to be generated by the computer.”

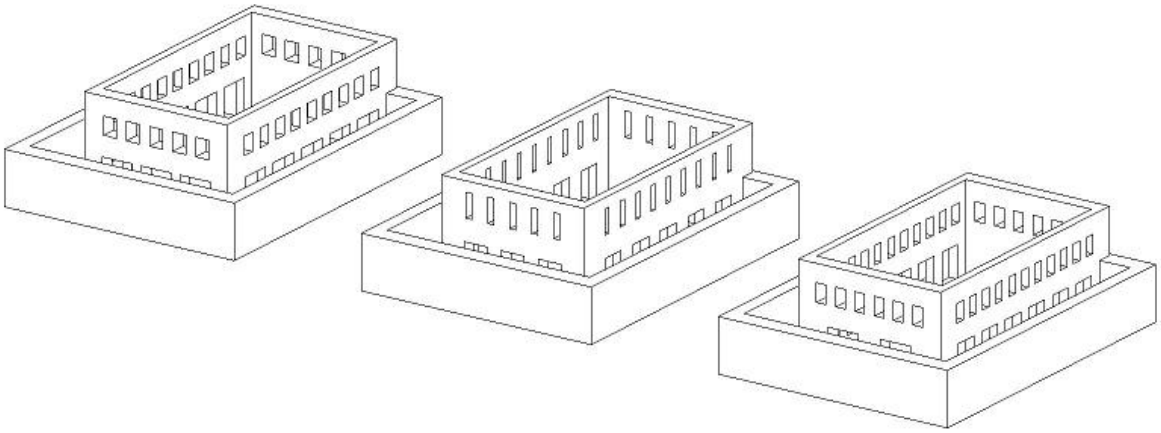


Figure 3.1 Alternative solutions

Brugman took up the challenge to focus entirely on the form, quality and perception of space. These issues are normally a mater of feeling, instead of mathematical output. Brugman looked for a design theory which made it possible to calculate the ideal form of space. This study let him to the design theory of Dom van der Laan. The design theory of Dom van der Laan is based on the so called ‘ideal’ measurements. Dom van der Laan reduced these to mathematical expressions. These expressions are the basis of the dimensions of his buildings. Brugman adapted these expressions and coded these in a script, in order to make ‘infinite’ alternatives, and to study these alternatives, figure 3.4. He used for his study the Monastery of Vaals, this is one of van der Laan’s building designed according to these expressions.

Figure 3.2 Rendering of the generated interior

One of the generated alternatives is rendered, figure 3.5, to have a more realistic view of the design. With use of a laser cutter the 3D model is transformed to a physical model. This project made clear what the strength is of parametric designing, in design studies.

3.2.1 Madolis (H.J. Bijlsma)

Differentiation and unification in interaction with each other, these are the underlying principles of the Silodam project in Amsterdam, designed by MVDRV. “The final design left many question unanswered, as if it was the easy way out of this fascinating concept. What would be the effect if we use modern computers to generate complex but attractive alternatives, with use of a set of newly developed constrains.”

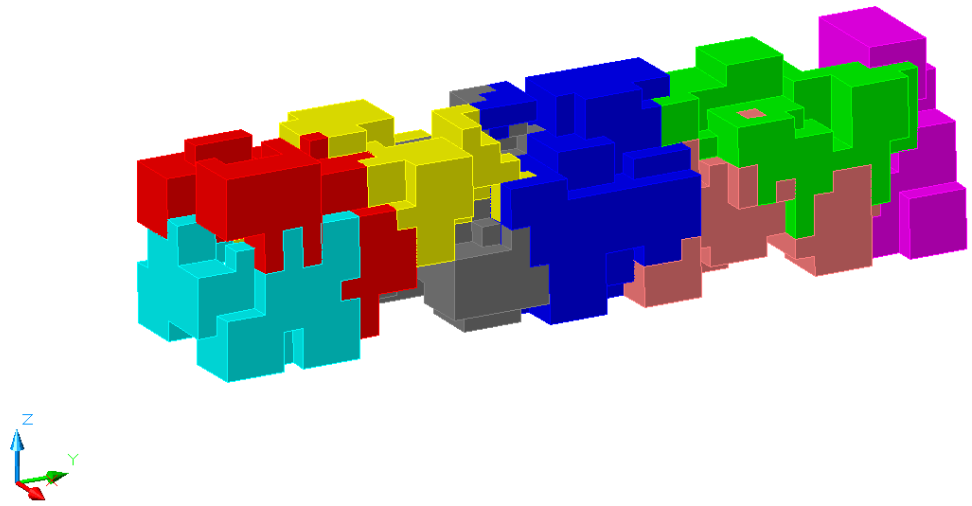


Figure 3.3 Basic Madolis

Madolis is an alternative solution to the Silodam project. The underlying concept is left intact, but the design is generated by the computer, figure 3.3. On the building site a pre-set number of buildings blocks is the starting point. A building block occupies a total floor area within certain limits. These building blocks will increase, with the smaller ones having a higher priority. Fulfilling the constraint area, the entree of each building block is created, to ensure that it is connected to the pier, figure 3.4.

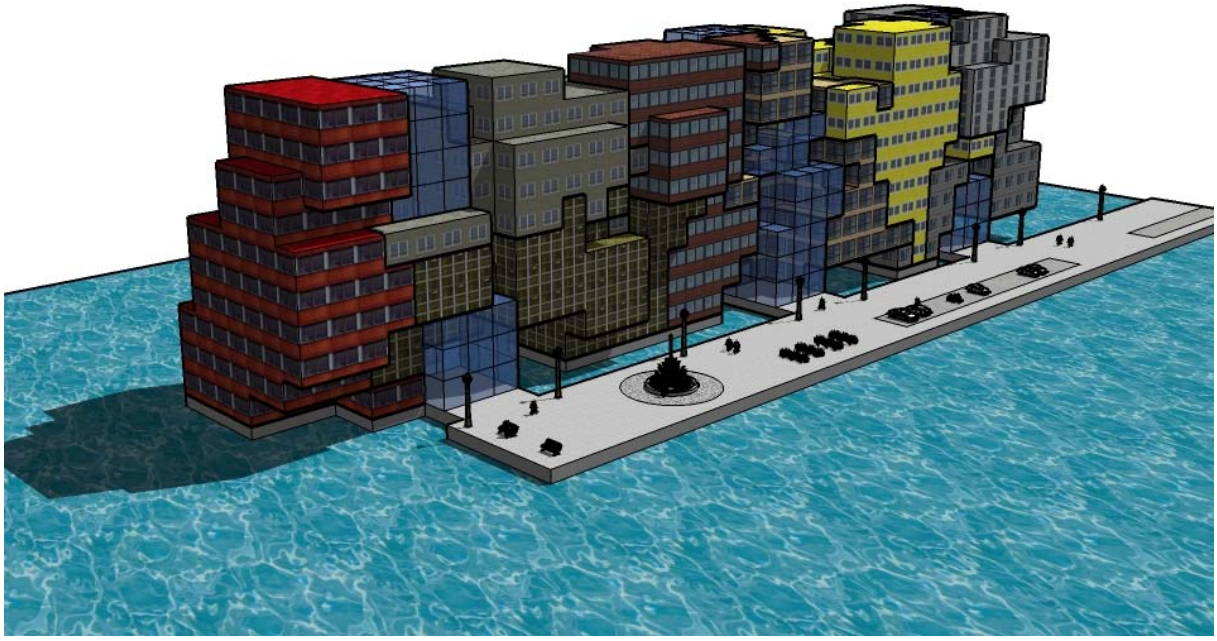


Figure 3.4 Rendered Madolis

The generated design is a complex but easily to comprehend building. The script, developed by Bijlsma, is based on cellular automata and shows that with use of simple rules complex buildings can be designed which has a resemblance with traditional designed buildings, figure 3.5.

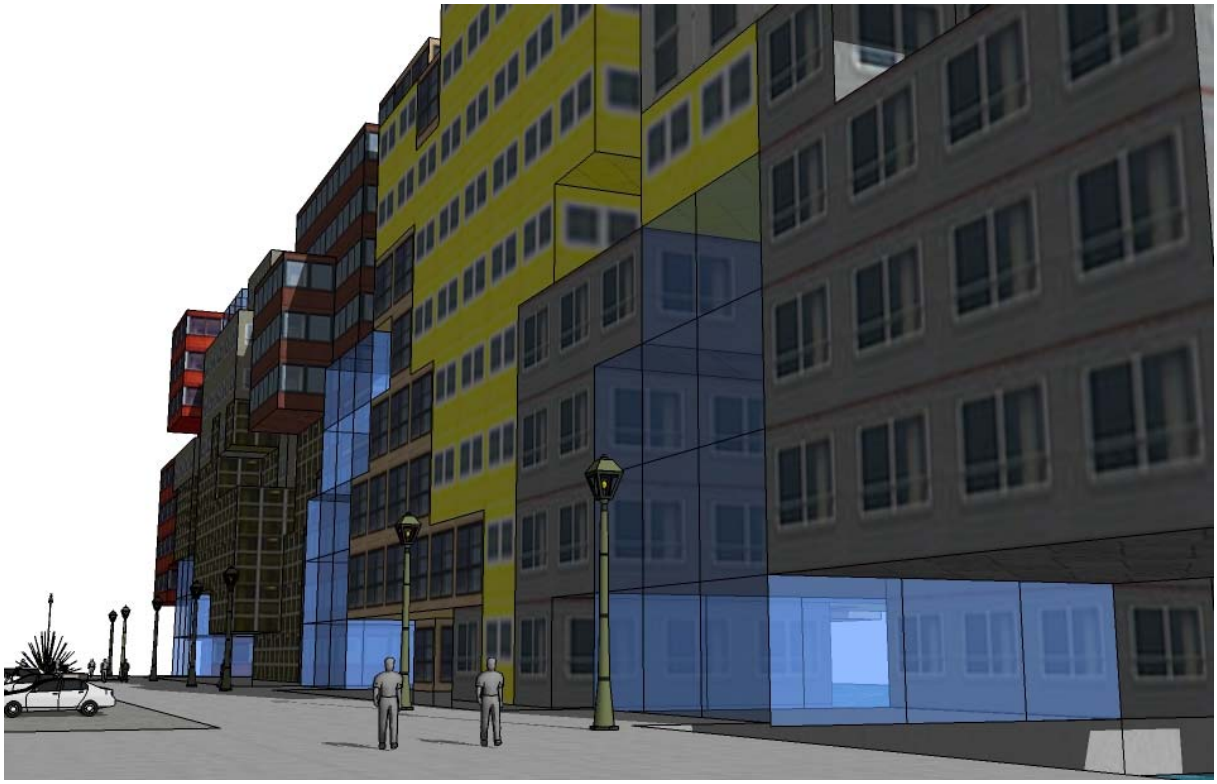


Figure 3.5 Walkthrough of Madolis

3.2.1 Tetris XX (B. Kramer)

Goal of this project was to develop a dynamic building which changed accordingly to the age of its habitants. Each habitant has to answer a number of questions about its age, minimal floor area, maximal width and length. The apartment will be ordered accordingly to age.

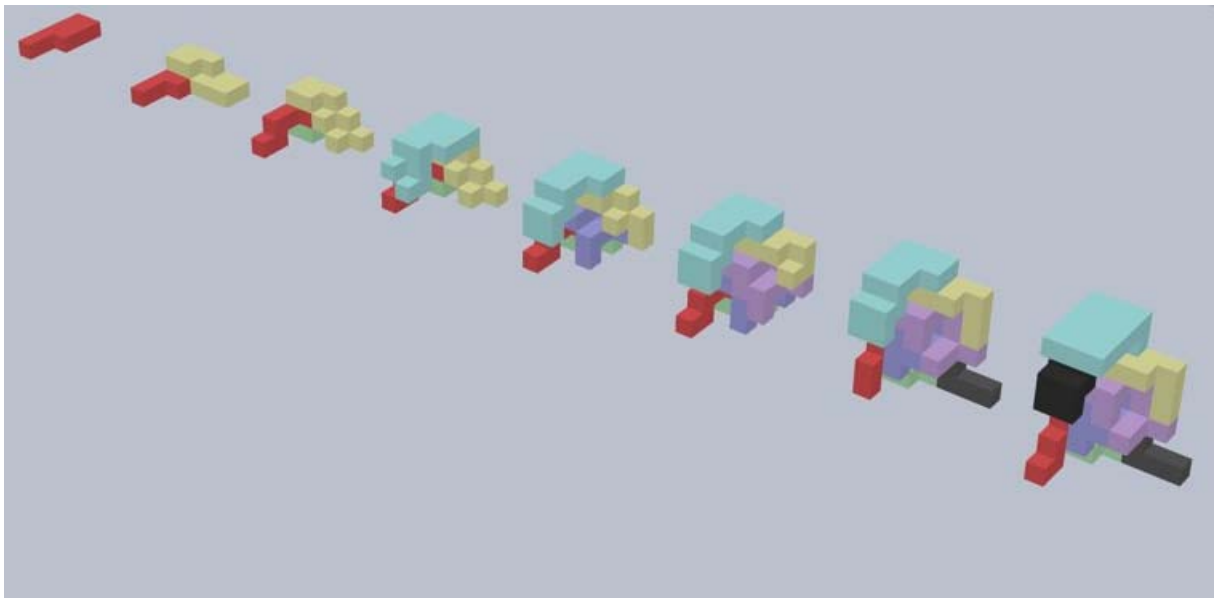


Figure 3.6 Time steps

This ordering will take place each time a new habitant moves into the building, see figure 3.6. The youngest inhabitant will find its apartment on the highest floors and the eldest on the lower floors. The younger are still vital and need their freedom. Families are living in the middle section; they are the separation between the younger and the elderly. They can keep an eye on the younger and lend a helping hand to the elderly. In course of time, families will be split up in elderly and younger. The elderly will be living close by the street, so they don't feel lonely or separated from the neighborhood. When entering the building the younger inhabitants are walking by the elderly.

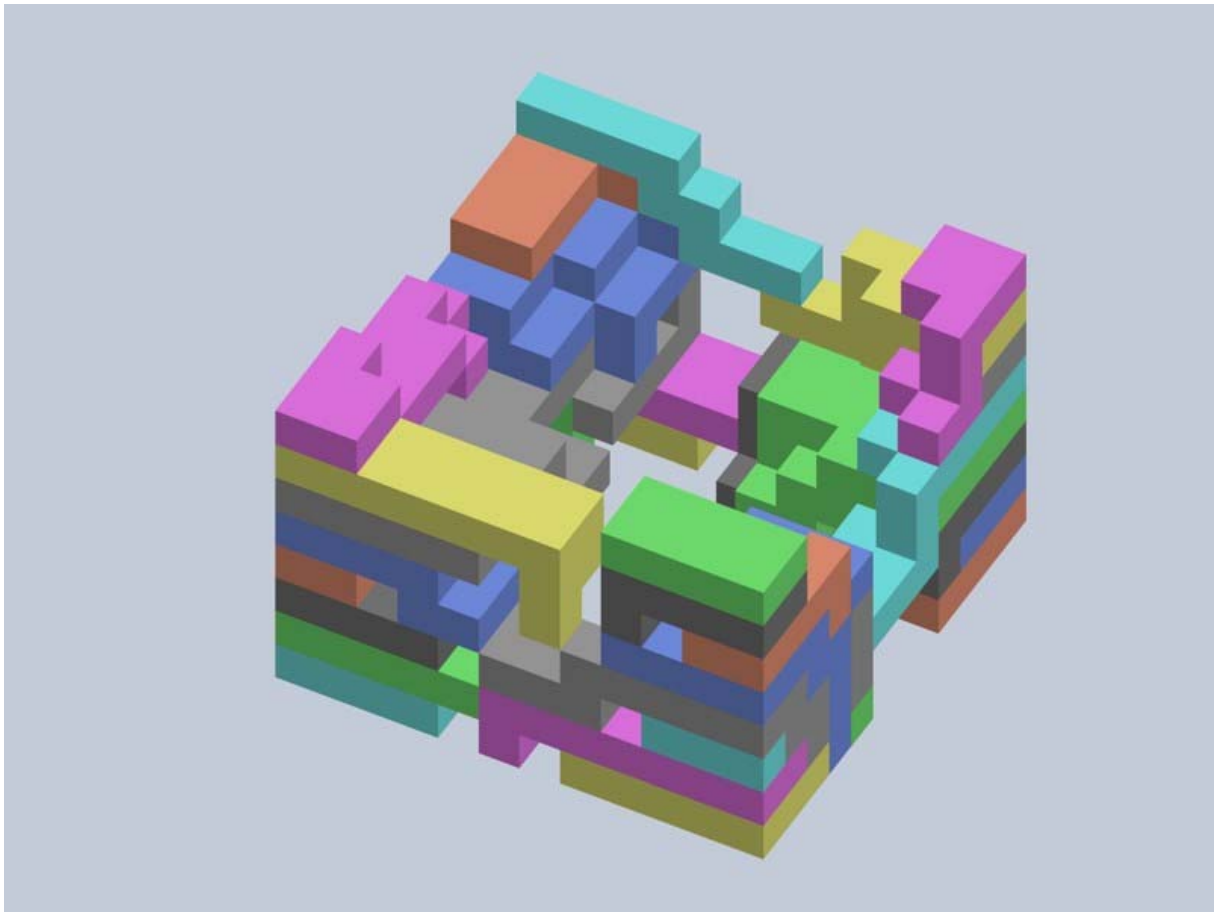


Figure 3.7 Apartment building

The script is based on cellular automata but with not so 'traditional' rules. If an inhabitant is older then 50 years their apartment will be on one floor, on the other hand if an inhabitant is younger then 50 their apartment the apartment will be occupy maximal 3 floors. Before the 'blocks' are moving one level downwards, there will be one block-unit left open, see figure 3.7. This space is an open-air terrace of the adjacent apartment.

3.2.1 Greedy Otto (G.J. Rohaan)

While generative designing isn't designing in the traditional way, it is necessary to find the parameters which determine the shape of the design. References are found in the surrounding nature. Variables as wind direction, sunlight, and minimal surface determine the final shape of for instance a tree. Goal of generative design isn't to imitate natural life forms, but to find the underlying ruling patterns. A general pattern is optimization and minimization. A so called minimal surface is a good example of this principle; every span has a shape in which the use of material is minimal. Extensive research in this domain has been done by Frei Otto, with use of soap bubbles. With use of this knowledge Frei Otto developed a way to create meshes which has a minimal distance between the vertices and in which the paths are minimized. The complex forms Frei Otto designed looks at first glance rather arbitrary, but are the result of the ruling pattern and prescribed parameters. Goal of this project is to develop a structure which at first glace is rather arbitrary but has

ruling pattern ‘optimization’. This project was a way to investigate how to convert the parameters into a script. The script has to create a structure in which the distance between the vertices is minimized as well the total length of the paths. Each path between the vertices has a weight factor; the result is that vertices can get more weight then others. Rohaan used the ‘force density’ method to calculate which vertices would be connected. The generated structure will react to changes in these weight factors by pushing away or attracting added vertices. The script generates a structure based on entered points. The resulting structure has no scale, and can be used for town planning problems as well for wall design. To illustrate this multi usability, the script is used to generate to solution on a different scale:

- Hallways: the script is used to generate an alternative layout for the hallway between the buildings on the campus, see figure 3.8.

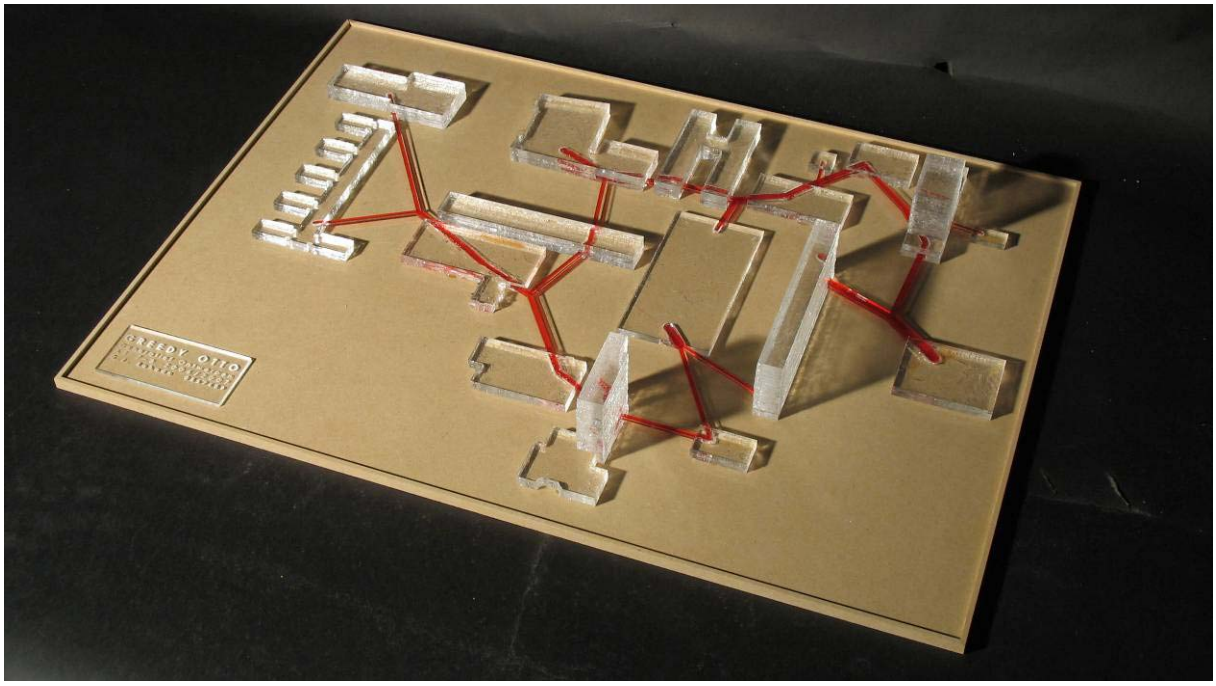


Figure 3.8 Hallways

- Façade: The façade is build made of 4 elements, each with a 6 by 6 dimension. Each element is made of 8 points, chosen in such way that the elements have a good connectivity. Within this framework lie 5 arbitrary points, so each element will have a different structure. The façade is the result of joining 4 sides together. Each side consists of a group of 6 elements which will be mirrored on both axes, see figure 3.9.

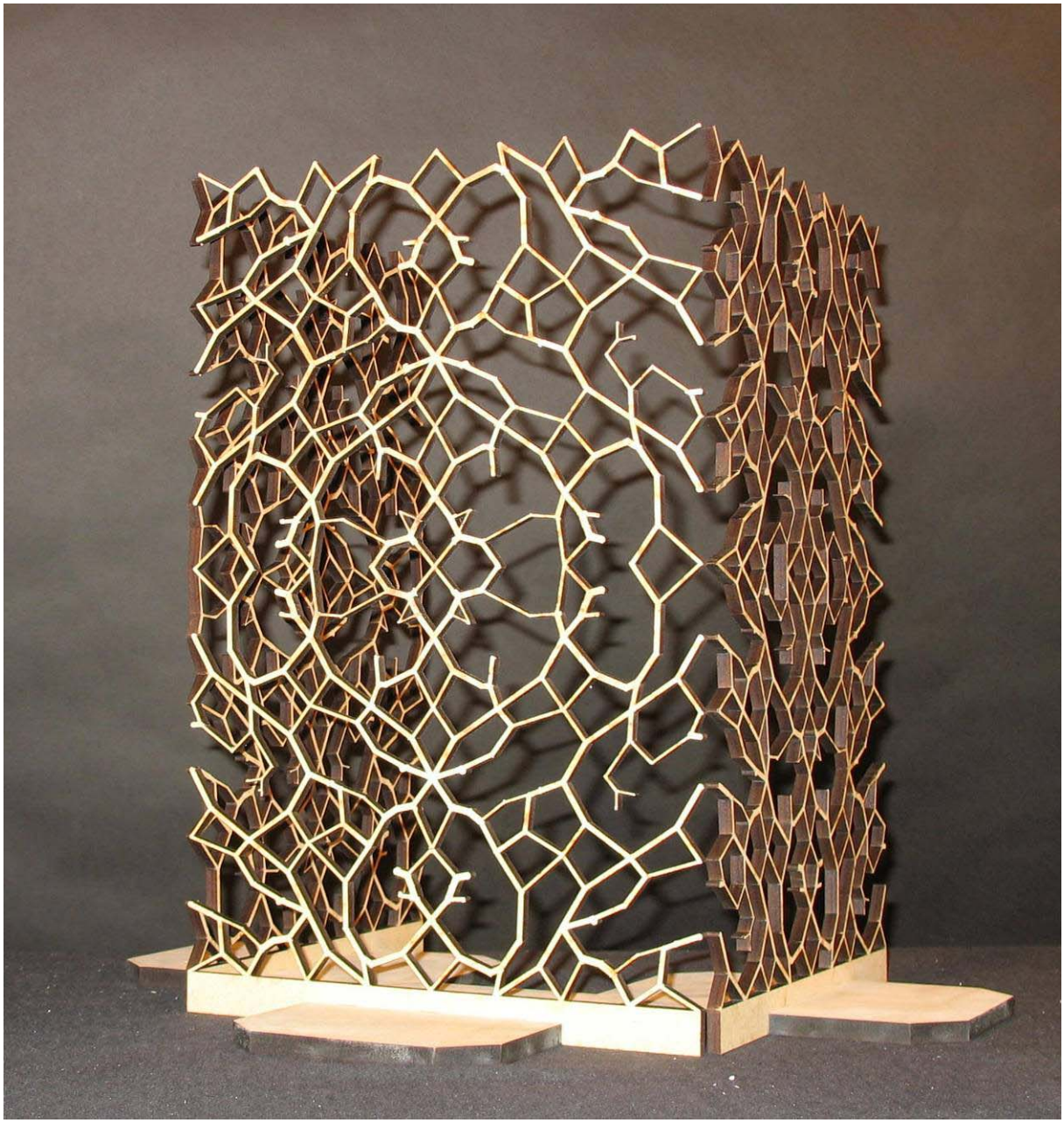


Figure 3.9 Façade

3.3 Conclusions

Students who participate in the design studio learn a new way of designing. They also learn that there is a hidden relation between many aspects of the design and they learn to think and look in a more abstract way.
By phrasing their own problem in the first week of the design studio the students are shaken up. For the first time in there study they don't have a ready made problem. The students are thrown back to their self. The second confrontation is learning to master a script language. After a slow start, the students learn quickly to program a script, and the end result is always astonishing.

By translating an, developed by their own, algorithm, with use of an embedded script language, into ‘a script’, the CAD-software can perform the task. According to Terzidis (2008, pp 65) “an algorithm is a computational procedure for addressing a problem in a finite number of steps. It involves deduction, induction, abstraction, generalization, and structured logic.” He continues with “Algorithmic strategies utilize the search for repetitive patterns, universal principles, interchangeable modules, and inductive links.” Every one of the former mentioned ‘actions’ is part of a general problem solving strategy. So learning to develop an algorithm from scratch as well learn how to program, serves a few purposes, namely:

- Learn to solve a problem in general way;
- Learn to master CAD software in a more fundamental way;
- Learn to surpass the limitation of the ‘out of the box’ CAD ?software;
- Looking at a problem in a different way.

After this scripting period the proto typing phase is a felt as a ‘back on home ground’. With the experience of two design studios we conclude that students adapt quickly to this new way of designing. Some students see the advantage of this way of thinking and the possibilities to generated new alternatives. But some students think it is a nice experience, but will probably not design like this again.

4. References

DeLanda, Manuel (2002) Deleuze and the Use of Genetic Algorithms in Art, In A. Rahim(ed) Comtemporary Techniques in Architecture (Architectural Design), jan. 2002

Duarte, J. (2000). Customizing mass housing: a discursive grammar for Siza's Malagueira houses. PhD-Thesis. Faculty of Architecture, Massachusetts Institute of Technology.

Galanter, P., (2003) What is Generative Art?, Complexity Theory as a Context for Art Theory, 6th International conference Generative Art, Milan, 10-13 Dec. 2003, pg 216-236, editor Celestino Soddu.

Holland, J. H. (1992) Adaptation in Natural and Artificial Systems; MIT-press, Cambridge, Massachusetts

Holland, J. H. (1992) Genetic Algorithms, Scientific American, 66-72

Kolarevic, Branko (2005) Architecture in the Digital Age, Taylor & Francis Group

Mitchell, (1990)

Terzidis, K. (2008) Algorithmic Architecture, Architectural Press

Zee, A. van der, Vries, B de (2003), Checking interactive generated design against distributed objectives. 6th International conference Generative Art, Milan, 10-13 Dec. 2003, pg 19-28, editor Celestino Soddu.

Zee, A. van der, Vries, B de (2004), Interactive Generated Design Alternatives Constrained by Technical and Spatial Conditions. Generative CAD Systems, proceedings of GCAD 2004 editors Omer Akin, Ramesh Krishnamurti and Khee Poh Lam. Pg 473-492.