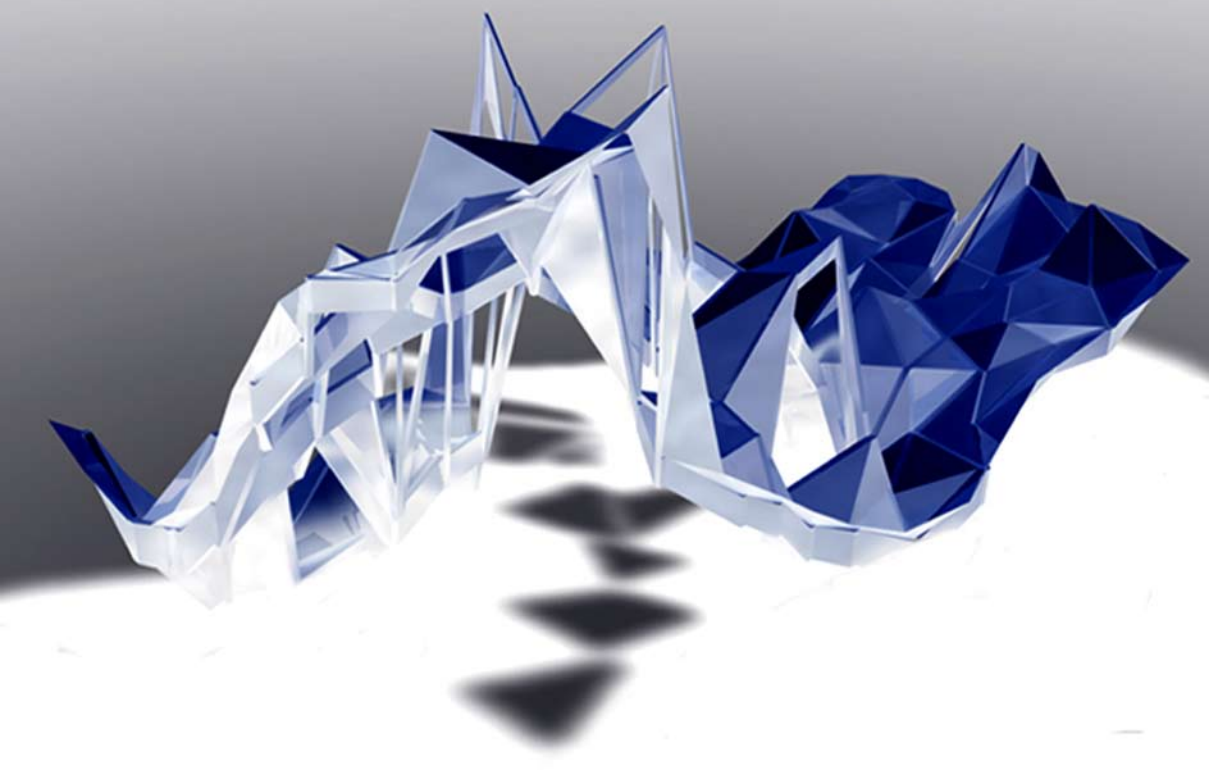
Algorithmic Extension of Architecture

Toni Kotnik



Algorithmic Extension of Architecture

Toni Kotnik

MAS ETH ARCH/CAAD
2005/06
Individual Thesis

Zurich
October 2006

dedicated to the memory of my father

Acknowledgement

This individual thesis is part of my postgraduate studies in computer-aided architectural design at the Swiss Federal Institute of Technology (ETH) Zurich. My thanks go to Prof. Dr. Ludger Hovestadt and his team at the chair of CAAD for their hospitality and cooperativeness and Philipp Schaerer for the smooth organization of the course. Part of this thesis was done during a stay at the Architectural Association London. For inspiring comments on my project I would like to thank Tom Verebes, Eugene Han and George Liaropoulos-Legendre.

Content

Abstract

Over the past decade an increasing amount of architects have readdressed formal issues. This development is closely related to the availability of powerful computers and software that enable the use of computational mechanisms for the exploration of formal systems. However, up to now the theoretical foundations of this new digital methods in design are still unformulated. The present work, therefore, tries to formulate an abstract conceptual framework for the evaluation of different digital approaches to architectural design. This framework is based on the Turing machine as an abstract model for the computer and it results in an algorithmic description of every task performed by the machine. As a consequence, every form generated in a digital design process is bound to an algorithmic description of its own morphogenetic process. The level of awareness of this relation is then used to formulate a conceptual framework of digital design. Furthermore, the framework is compared to a similar approach to the digital in architectural design by Oxman. Both studies show, that the digital way of designing and exploring architecture has to be seen as an extended form of expression that is interwoven with the non-digital in manifold ways. Finally, the design of a roof structure gets used to examine the algorithmic approach to architecture in more detail.

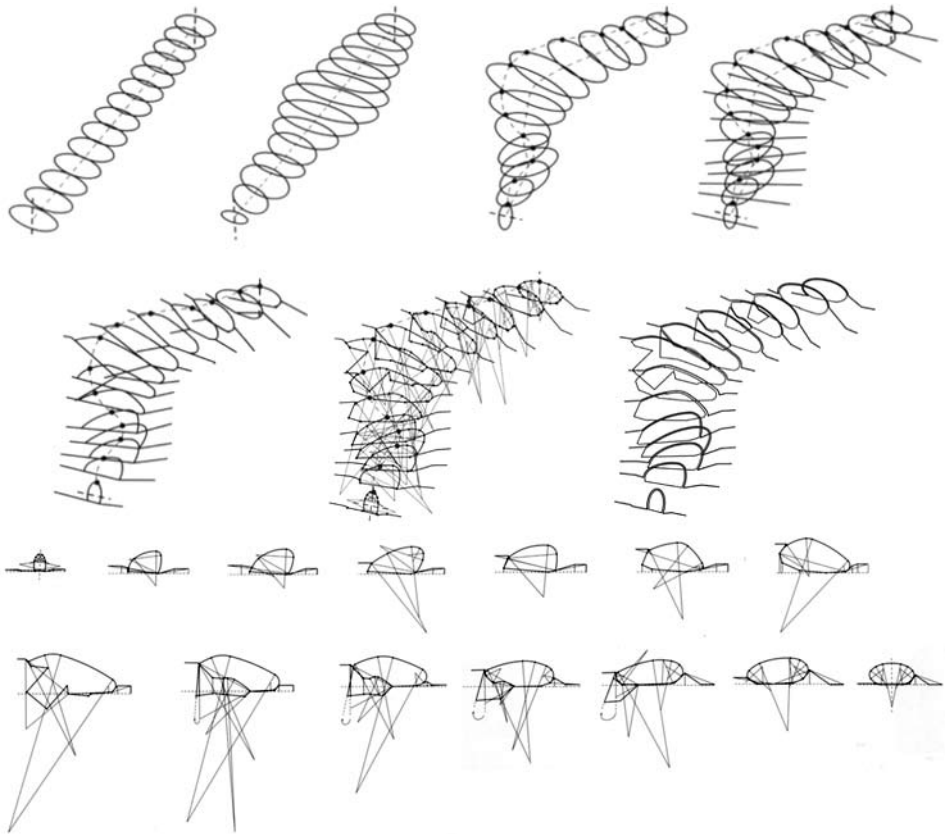
Theory

Coop Himmelb(l)au: UFA Cinema, Dresden, Germany, 1996-98
shifting and shearing as manageable geometric operations

In 1963 Ivan Sutherland's Sketchpad program, the first interactive graphical design tool, demonstrated that computers could be used for drafting and modeling, not only for number crunching, and already by the mid-1990s architectural practice without graphics software had become unimaginable. However, in spite of the fact that computer-aided design technology has been adopted almost universally as the predominant means of production in architectural practice, its use merely represents the commercialization of the simplest and most obvious application of information technology in architectural design - the automation of traditional processes like drafting, modeling, and communicating - without adding value to the practice and its products. [Kalay, 2004:xvi] As a result, most architectural design solutions are still crafted manually, much the same way they have been for the past 500 years.

> *"Beyond the fact that over the past decade a new generation of avante-garde architects is pushing digital technology to its limits, so far it has had relatively little qualitative impact on the profession of architecture at large. In general, information technology has improved the efficiency of designing buildings, when in fact it has the potential to reinvent the architectural design process itself." [Kalay, 2004:xvi]*

In the present work the goal is to formulate an abstract theoretical framework for digital design in order to activate this potential for the reinvention of the design process by means of the computer. Up to now, the theoretical foundations of digital design are still unformulated with the basic concepts bound up in ideological positions. [Oxman, 2006:239] In avoidance of such struggle, this paper is based on the simple observation that the possible reinvention of architectural thinking in the digital realm is essentially bound to the computer as its principal design tool.



Lars Spuybroek: Water-experience pavilion, Neeltje Jans, Netherlands, 1993-97
modeling of beam structure as concatenation of circular segments

In general, every tool has a specific way it can be used and this peculiar form of usage influences the perception of the user and his way of thinking. That is why the computer itself should be the starting point of an investigation into the digital design of architecture. Using methods from computability theory a view onto the discipline from the outside will be achieved that enables a better understanding of the way the design process will be influenced by the digital. Furthermore, this distancing from the discipline of architecture avoids any ideological positioning inherent to every discussion from within.

Computability

The versatility of a computer is built on the universalitiy of its main principle: a machine, the hardware, manipulating data accordings to a set of instructions, the software. In this general setting every data takes the form of a finite sequence of bits and that is why it can be coded as a natural number. Hence, a program **p** can be viewed as partial function on the set of natural numbers $\mathbb{N}$ with output **out** $\in \mathbb{N}$ as result of a computation of the input **in** $\in \mathbb{N}$ that is **p**(**in**) = **out**. (Figure 1)

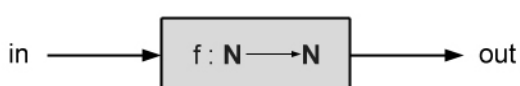


Figure 1: functional description of program

It is this abstract setting for a computing device that facilitates the principal question of computability, i.e. for which (partial) function **f** exists a program **p** such that **f**(**in**) = **p**(**in**) for every valid input **in** $\in \mathbb{N}$. Surprisingly, this question existed already before the invention of modern digital computers. In the 1930's various mathematicians like Alonzo Church, Kurt Gödel or Alan Turing started to develop precise, independent definitions of what it means to be computable in order to

Jakob & MacFarlane: Restaurant Le Georges, Centre Georges Pompidou, Paris, France, 1999
topological deformation of regular grid structure

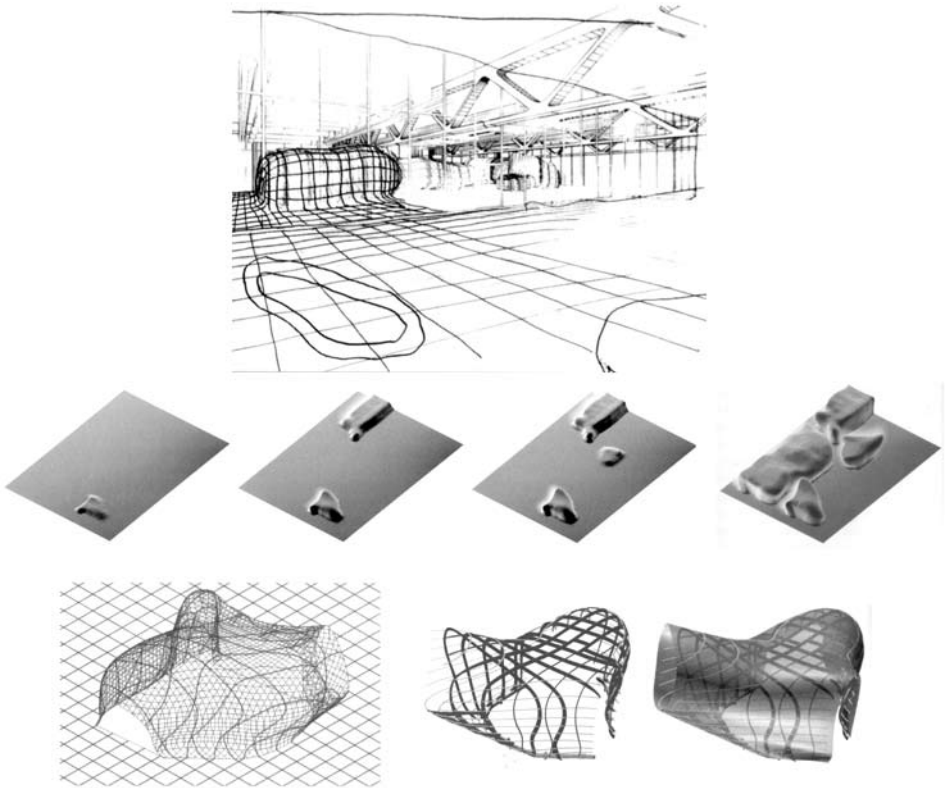
answer a mathematical problem stated by David Hilbert.

Intuitively, a task is computable if one can specify a finite sequence of instructions which when followed will result in the completion of the task. This intuition must be made precise by defining the capabilities of the machine that is to carry out the instructions, because machines with different capabilities may be able to complete different sets of instructions and, therefore, may result in different classes of computable functions. However, it is a remarkable mathematical fact that all the different precise definitions of computability lead to the same class of functions. In a practical sense this means that if one can give an intuitively convincing description of an algorithm for computing a function, than one can find an effective procedure in one of the precise definitions as well.

Turing machine

In 1936 Alan Turing published a paper on Hilbert's problem. [Turing, 1936] A by-product of this mathematical work was the first machine-based model of what it means for a function to be computable, and the description of what is no called a Turing machine. These simple abstract devices are one of the earliest and most intuitive ways to make precise the intuitive idea of computability and the underlying logic is closely connected to the later development of computers.

A Turing machine consits of an infinite one-dimensional tape divided into cells, a movable read-write head with a specified starting position, and a table of transition rules. (Figure 2) Each cell of the tape contains one symbol, either 0 or 1, and the head can move along the tape to scan one cell at a time and perform three different activities:

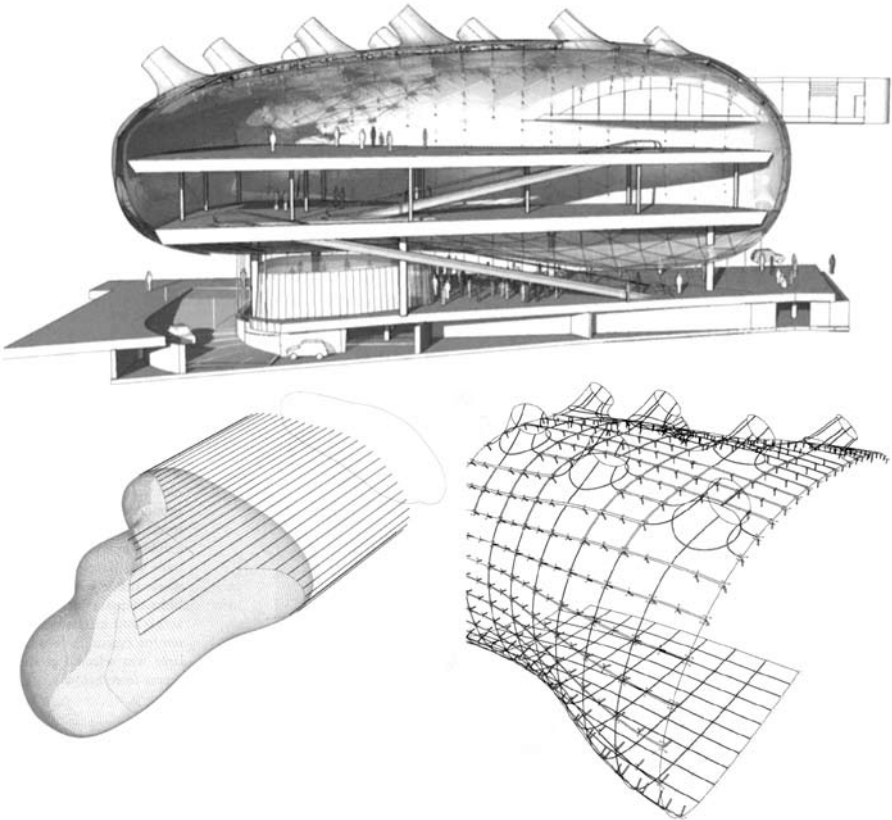


Jakob & MacFarlane: Restaurant Le Georges, Centre Georges Pompidou, Paris, France, 1999
topological deformation of regular grid structure

- READ: read the content of the cell,
- WRITE: change the content into the opposite, and
- MOVE: advance to the next cell to the right or left along the tape.

A table of transition rules serves as the program for the machine. Each such rule is a quadruple <$state_{actual}$,symbol,action,$state_{next}$ > which means if the machine is in $state_{actual}$ and the current cell contains symbol then take action MOVE or WRITE and move into $state_{next}$. Thus, the transition rules are labeled as $state_n$ and the execution of the program consits of the successive transition between one state and another. Furthermore, the program terminates if it reaches a situation in which there is not exactly one transition rule specified for execution. [Barker-Plummer, 2005; Cooper, 2004:34-42]

Figure 2: visualization of Turing-machine



Peter Cook & Colin Fournier: Kunsthaus, Graz, Austria, 2002-03
NURBS-modeling of skin

Figure 3: algorithmic description of program

Turing machines are very basic but powerful devices. They are not physical objects but mathematical ones and, despite theit simplicity, enable thought experiments about the limits of mechanical computation because they can be adapted to simulate the logic of any software that could possibly be constructed. In other words, for every partial function **f** defined by a program **p** there exists a Turing machine **T** with **f**(**in**) = **T**(**in**) for every input **in** $\in \mathbb{N}$. (Figure 3)

Algorithmic perspective

Without exception, using a computer always means to activate an algorithmic procedure as mediator between input and output. Therefore, the abstract model of the Turing enables the shift of awareness from the machine towards the principle represented by the machine. As Kalay has pointed out before, the mere use of computers in architecture does not change the way of production. What is needed is a conscious consideration of the machine in order to be able to reinvent the design process in architecture and use the computer creatively. This is, what the Turing model stands for. In addition, what is needed is a conscious differentiation between computation and computerization.

> *“While computation is the procedure of calculating, i.e. determining something by mathematical or logical methods, computerization is the act of entering, processing, or storing information in a computer or a computer system. Computerization is about automation, mechanization,*

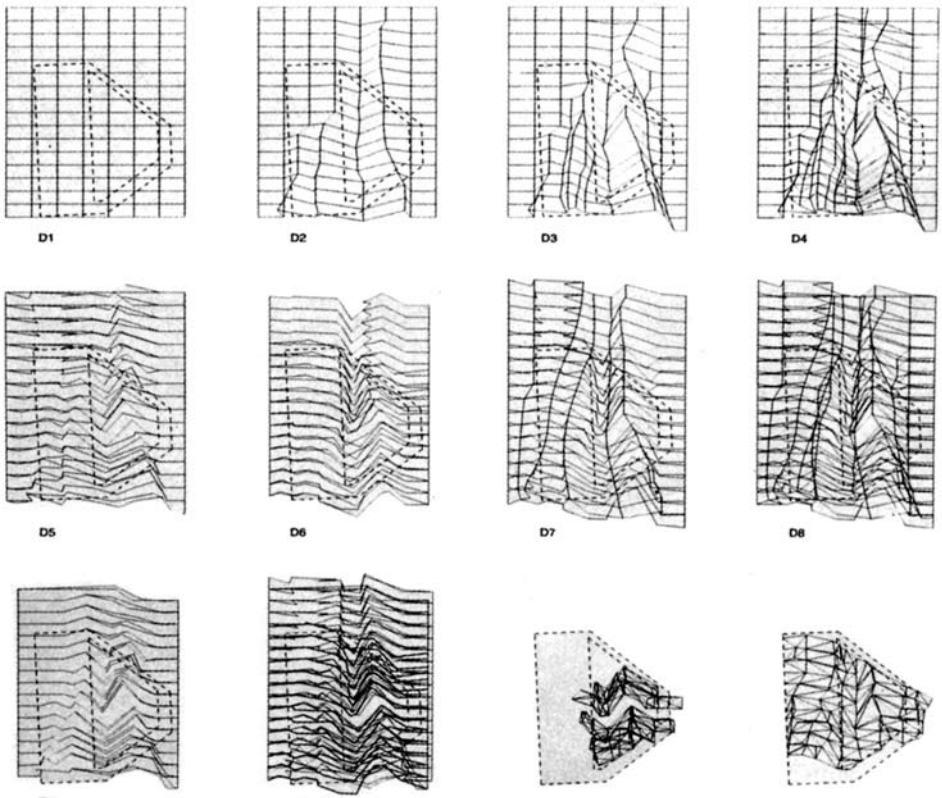
Peter Cook & Colin Fournier: Kunsthaus, Graz, Austria, 2002-03
NURBS-modeling of skin

> *digitization, and conversion. Generally, it involves the digitization of entities or processes that are preconceived, predetermined, and well defined. In contrast, computation is about the exploration of indeterminate, vague, unclear, and often ill-defined processes; because of its exploratory nature, computation aims at emulating or extending the human intellect. It is about rationalization, reasoning, logic, algorithm, deduction, induction, extrapolation, exploration, and estimation. In its manifold implications, it involves problem solving, mental structures, cognition, simulation, and rulebased intelligence, to name a few." [Terzidis, 2006:xi]*

Because of this very close relation to human endevours the question of computation existed already long before there were any computers and is rooted in mathematics of antiquity. To foster algorithmic thinking in architecture, therefore, not only is a way to utilize computers in the design process productively but might be the key concept to develop a critical theory of digital architecture. A first step towards such a theorizing, therefore, would be a stocktaking of contemporary architecture from an algorithmic perspecitive.

As a Turing machine every algorithm defines a partial function **f** on the natural numbers with a table of transition rules <.,.,.,.> as functional description. The domain of **f** that is the set of valid input is the parameter space **Para** $\in \mathbb{N}$. The image **f**(**Para**) of the parameter space is the set of possible variations **Var** $\in \mathbb{N}$ in the output of the Turing machine. (Figure 4)

In an architectural design context, an element **out** $\in$ **Var** represents the coded version of a form gradually generated in a digital environment, e.g. a CAD-software. In other words, the abstract Turing-based machine model can be used in theory to provide a formal algorithmic description of the morphogenetic process of design. And it is this formalization of architectural thinking that will be used as



Peter Eisenman: Chruch of the Year 2000, Rom, Italy, 1996 (competition)
morphing of regular grid into superimposed diagram and three-dimensional interpretation of result as system of folding

measure in order to achieve a first rough classification of the utilization of the computer in contemporary design processes thereby focusing on the generation of form.

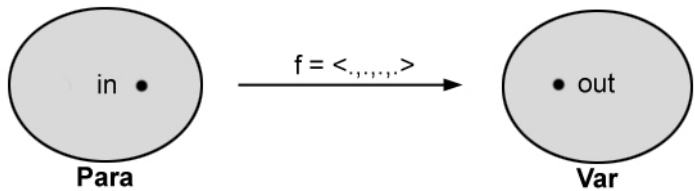
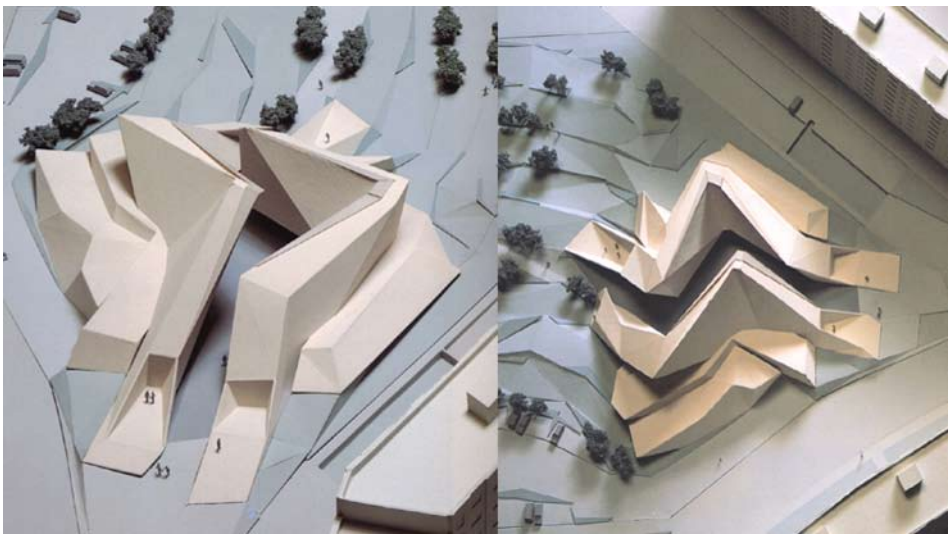


Figure 4: space of parameters and space of variation related to program

Drafting and modelling as graphical computation

Today, the dominant use of computers in architectural design still is only as an efficient tool of representation of form through drafting and modelling. Thereby, within the realm of the peculiarities of the deployed CAD-software a digital model of the architecture gets build up inductively using primary forms and a set of different possibilites of modification of these forms.

In general, all the geometric information in a computer-aided environment is based on the use of non-uniform rational Bezier splines (NURBS). [Piegl & Tiller, 2000] A NURBS is a smooth curve from a startpoint A to an endpoint B defined through a set of attracting control points P_i and corresponding weight functions w_i, regulating the degree of attraction. Thus, every NURBS is the graphic representation of an output achieved as result of a computation based on the input of the startpoint, endpoint, control points, and weight functions. That is, every production of a NURBS is an algorithmic transformation of an input, controlled by mouse and keyboard, into a graphic output. Therefore, it can be seen as an activation of



Peter Eisenman: Chruch of the Year 2000, Rom, Italy, 1996 (competition)
morphing of regular grid into superimposed diagram and three-dimensional interpretation of result as system of folding

a specific Turing machine inherent to the used software that is modelling the computation of the curve. (Figure 5)

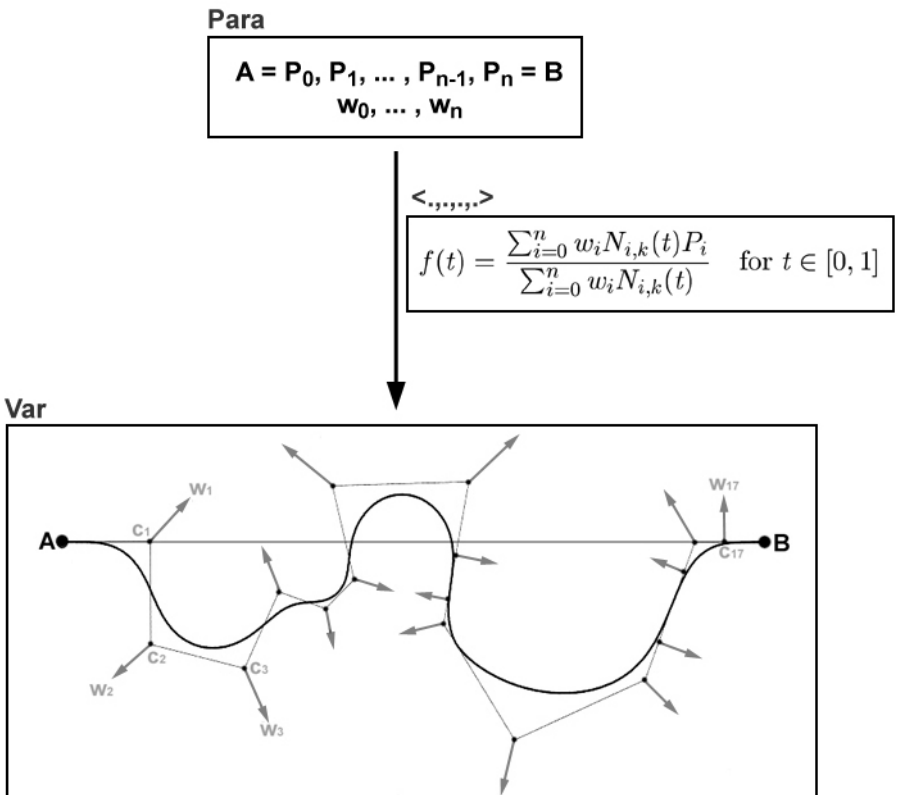
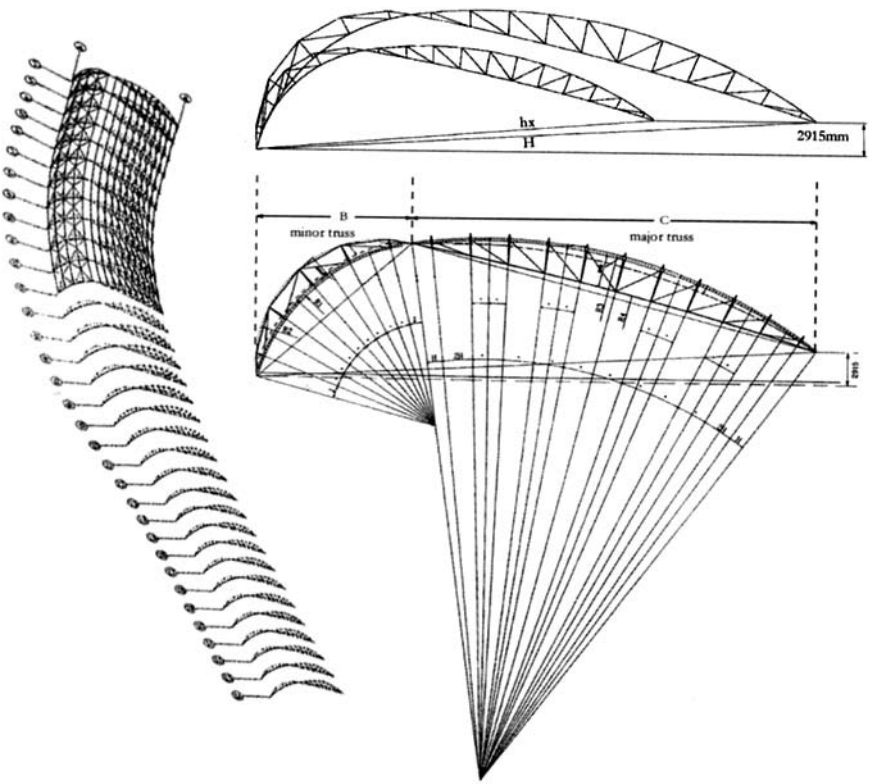


Figure 5: program structure of NURBS-calculation



Nicholas Grimshaw and Partners: Internationl Terminal, Waterloo Station, London, UK, 1993
parametric defintion of truss geometry with dependency on breadth h_X of railroad

This example shows that as tought experiment one can view the complete CAD-software as a finite collection of Turing machines $\{\mathbf{T}_1^{cad}, .. , \mathbf{T}_m^{cad}\}$, each one mediating between possible inputs of the user and consequential graphical outputs displayed on the screen. Thereby, every available tool defines a different Turing machine $\mathbf{T}_i^{cad}$. Because of this, the inductive process of drafting or modelling architecture by means of CAD-software implies the successive use of a finite amount of Turing machines $\mathbf{T}_i^{cad}$ and, hence, defines a unique sequence of Turing machines $(\mathbf{T}_1^{proj}, .. , \mathbf{T}_n^{proj})$ related specifically to the project. (Figure 6) This concatenation of machines generates as a whole a Turing machine $\mathbf{T}_{proj}$ with input as sum of all inputs, i.e. $\mathbf{in}_{proj} = \mathbf{in}_1^{proj} \cup \ldots \cup \mathbf{in}_n^{proj}$, and the displayed final model $\mathbf{out}_{proj} = \mathbf{out}_n^{proj}$ as output.

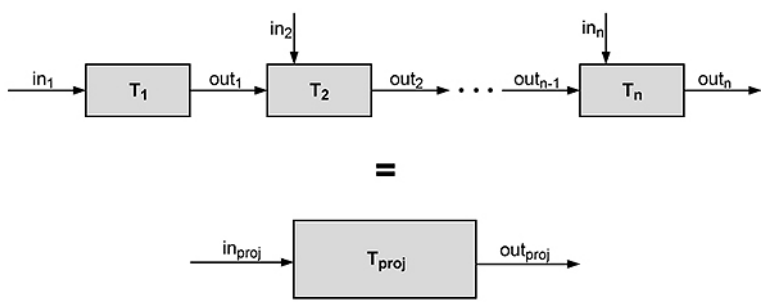


Figure 6: Turing-model of drafting/modeling

This means, even the mere use of the computer for drafting or modelling of architecture leads inevitable to an algorithmic description of the project. However, this description is not perceivable because it is hidden by the applied software and its ready-made geometric operations. Such an unconscious computation of architecture prevents an exploration of the inherent space of parameters or of the program itself in the realm of the design process.

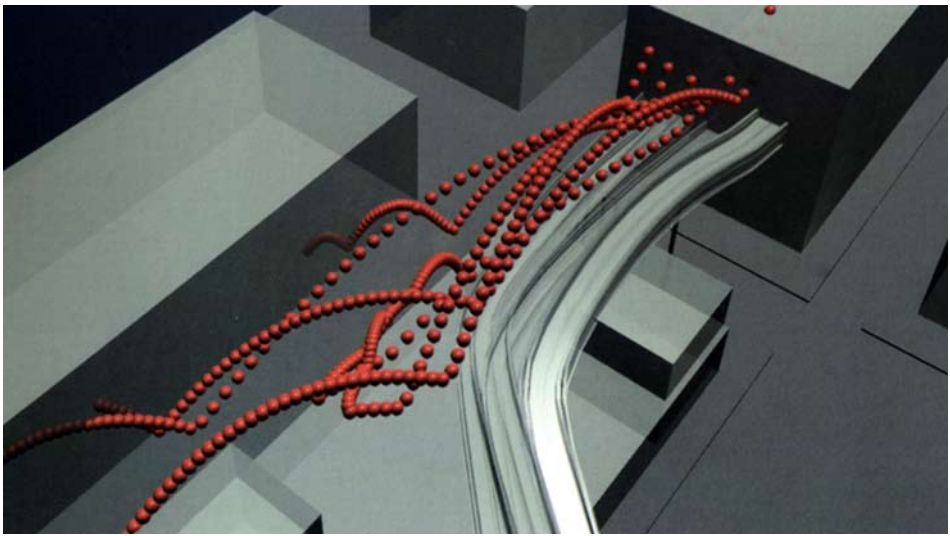
Bernhard Franken: Bubble, Frankfurt, Germany, 1999
form as metaball simulation of water drops

On the contrary, drafting and modelling is the reduction of the parameter space onto one fixed value, the specific input $\mathbf{in}_{proj}$, and with it the blinding out of possible variations. It is a static description of form from the outside that is a description of architecture based on ideas independent of the computers capabilities as essential design tool. That is why such use can be seen as purely representative. It is architectural design crafted in a traditional way, a waste of the possibilities inherent to the use of the computer in design.

From the representative to the algorithmic

Therefore, the threshold to digital design in architecture can be defined as the conscious overcoming of the traditional level of representation in the use of the computer as design tool. Thereby, looking at the development in architecture since the 1990s one can distinguish three degrees of computational awareness in this process of acquisition of the machine into architectural design: the operative, the parametric, and the algorithmic. (Figure 7)

On the operative level, the computer gets used for modelling in a pre-defined geometric way. That is implemented geometric operations of the software in use are explored in an architectural context in order to deform the classical formal language of architecure by means of controlled transformations. What distinguishes the operative from the representative is the type of geometric operations used for modeling, for example the shifting and shearing of a box in the UFA Cinema by Coop Himmelb(l)au (p.12) or the concatenation of circular segments in the process of meodeling the frame structure of the water pavilion by Lars Spuybroek. (p.14) Without a computer, such operations were not used widely in architecture because of the inherent increase of geometric complexity which limits the ability to handle them in an efficient way by means of drawing as



Greg Lynn: Port Authority Gateway, New York, USA, 1995 (competition)
visualization of database on density of traffic over time as animated particle; transformation of phase portrait into structural system of canopy

well as imagination. In the case of the operative, it is the computational power of the computer that opens up a new field of geometric possibilities in modeling. This gets obvious in the architectural use of the curvilinear language of contemporary CAD-softwares, the NURBS-geometry, like in the Kunsthaus Graz by Peter Cook and Colin Fournier (p.20/22) or the design of the restaurant in the Centre Pompidou by Jakob & MacFarlane. (p.16/18)

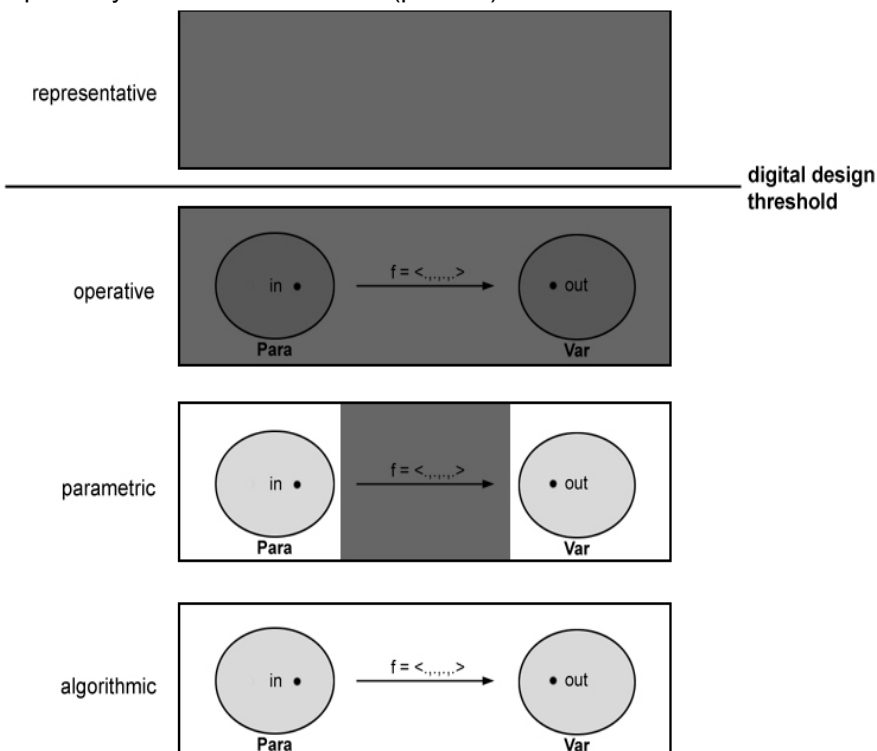
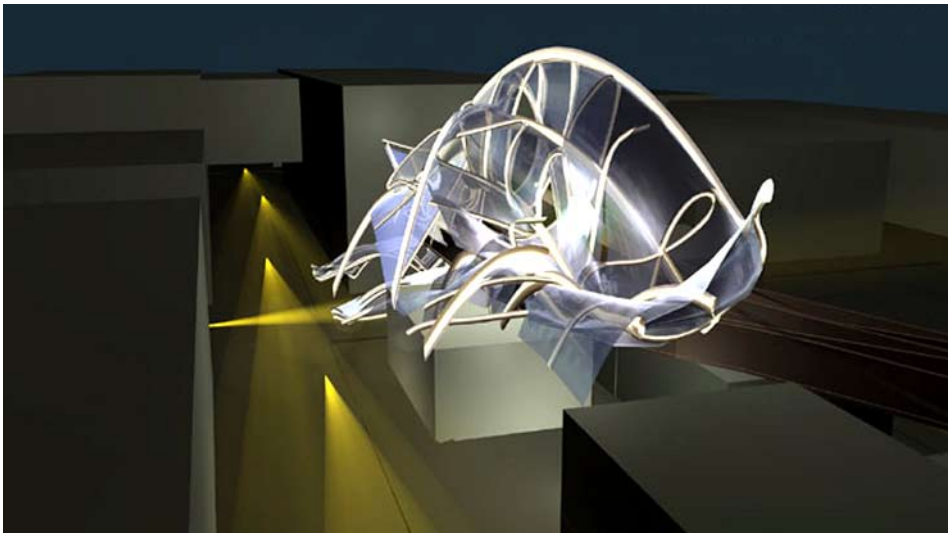


Figure 7: levels of algorithmic awareness

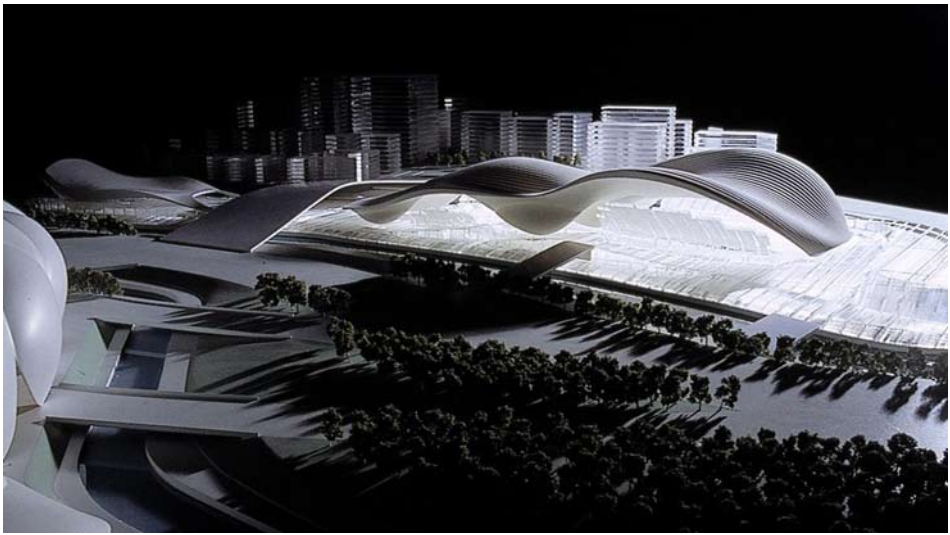


Greg Lynn: Port Authority Gateway, New York, USA, 1995 (competition)
visualization of database on density of traffic over time as animated particle; transformation of phase portrait into structural system of canopy

Especially the closer examination of the NURBS-geometry has fostered the parametric awareness and has helped to shift interest away from drafting and modeling towards a more mathematically based view on architectural design. This has to do with the fact that every NURBS object is defined within a local space of parameter given by control points and weights. These datas are not fixed but can be changed throughout the whole design process. That is why “parametrics can provide for a powerful conception of architectural form by describing a range of possibilities, replacing in the process stable with variable, singularity with multiplicity.” [Kolarevic, 2003:17] The design of thirty-six dimensionally different but identically configured three-pin bowstring arches for the International Terminal of Waterloo Station in London by Nicholas Grimshaw and Partners is an example for this parametrized variation. (p.28)

But by far the most popular way of using parameters in contemporary architecture is the utilization of time as primal parameter. Time-based techniques as morphing, keyframe animation, kinematics, force fields, or particle systems are widely used in the design process nowadays and are all based on the idea of gradually deforming a given NURBS-geometry by changing the parameters over time. Examples for this approach are the pavilion by Bernhard Franken (p.30), Greg Lynn’s project for the Port Authority Gateway (p.32/34), or the Aquatic Center by Zaha Hadid. (p.36)

As the Turing model shows, the strength of the computer as device is the flexible series of commands and logical procedures that can instantly transform it from one function to another. However, on the operative as well as one the parametric level, architects are forced to conduct the process of design using fixed Turing machines originally developed to solve the problems faced in different areas of use, for example in aircraft design or film-making. [Silver, 2006:9] Therefore, over the last years many architects have turned to the inhouse creation of code



Zaha Hadid: Aquatic Center, London, UK, 2005-09
geometry of roof out of simulation of behaviour of fluid by means of animated particles

appropriate to their specific needs. Only this step towards the algorithmic decription of the design made projects like the British Museum Great Court Roof (p.38/40) by Norman Foster and Partners, the Serpentine Gallery Pavilion (p.42/44) by Toyo Ito, the architecture for the Olympic Games in Beijing by PTW (p.46/48) and Herzog & de Meuron (p.50/52), or Ocean North's design for the Music and Art Center (p.54/56) possible. [Szalapaj, 2005:60-84; Balmond, 2004]

Towards a conceptual framework

Of course, the above description of a line of development in architecture with respect to the acquisition of the algorithmic into the design process has to be seen as a very rough first sketch. However, already this simplified picture of recent history in architecture elucidates the usefulness of the framework as a point of departure for a more detailed analysis of the usage of the computer in contemporary architectural design.

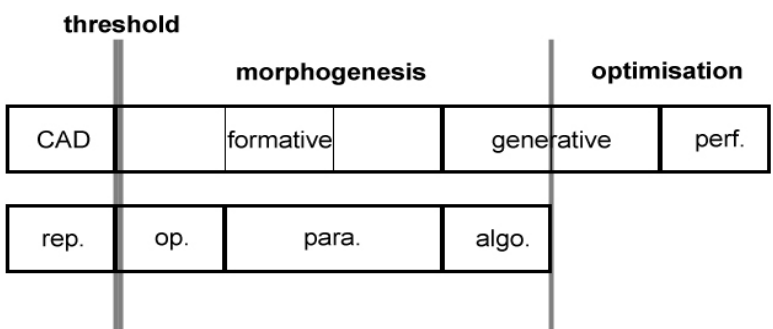
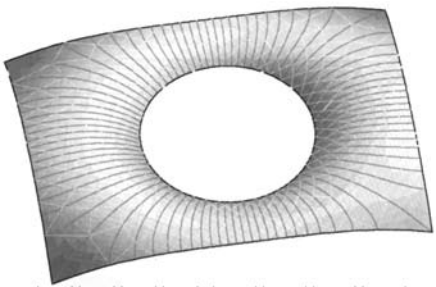
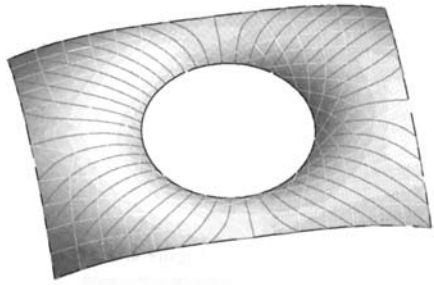


Figure 8: comparison of Turing-based model with Oxman's classification

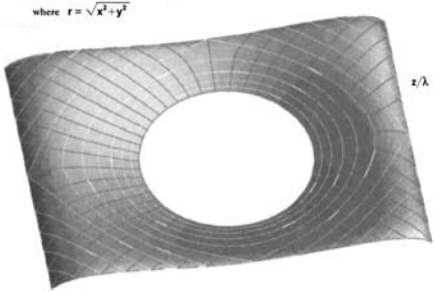


$$z/h=\left(1-x/b\right)\left(1+x/b\right)\left(1-y/c\right)\left(1+y/d\right)/\left(1-ax/rb\right)\left(1+ax/rb\right)\left(1-ay/rc\right)\left(1+ay/rd\right)$$

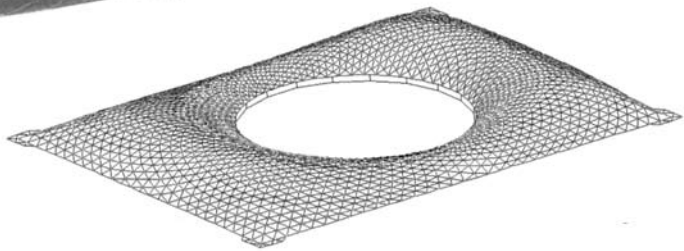
where $r=\sqrt{x^2+y^2}$



$$z/H=\left(1-x/b\right)\left(1+x/b\right)\left(1-y/c\right)\left(1+y/d\right)\left(\sqrt{x^2+y^2}/a\ -1\right)$$



$$z/\lambda\ =\ \left(\sqrt{x^2+y^2}/a\ -1\right)\Bigg/\left[\begin{array}{l}\sqrt{\left(b-x\right)^2+\left(c-y\right)^2}/\left(b-x\right)\left(c-y\right)\ +\\ \sqrt{\left(b+x\right)^2+\left(c-y\right)^2}/\left(b+x\right)\left(c-y\right)\ +\\ \sqrt{\left(b-x\right)^2+\left(d+y\right)^2}/\left(b-x\right)\left(d+y\right)\ +\\ \sqrt{\left(b+x\right)^2+\left(d+y\right)^2}/\left(b+x\right)\left(d+y\right)\end{array}\right]$$



Norman Foster and Partners: Great Court Roof, British Museum, London, UK, 1999-2000
geometrically defined parametric model based on algebraic overlay of three surfaces; shape optimisation by method of relaxation

Similar approaches to the digital can be found in the work of Oxman and her attempt to formulate a theoretical basis of design in the ‘first digital age’. [Oxman, 2006] Her study is based on the nature of interactivity and type of control of design processes and results in a comparable subdivision, namely „five paradigmatic classes of digital design models: CAD models, formation models, generative models, performance models, and integrated compound models“. [Oxman, 2006:246]

Not by accident, there are some similarities to the suggested conceptual frameworks. (Figure 8) The approach of the present work is founded on the degree of conscious perception of the abstract Turing-model for computers. And it is the resulting awareness of the algorithmic nature of this model that gets reflected in the level of interactivity and the degree of complexity of the processes controlled by the machine. That is why the representative level of the Turing-based model is congruent in substance to the descriptive CAD-model of Oxman. Furthermore, both approaches locate the threshold of digital design right above this level of interaction. [Oxman, 2006:260-262]

In the further subdivision of the field of digital design, however, there are some noticeable differences. Oxman’s second class of formation models consists of three sub-classes: the topological formation models, the associative design formation models, and the motion-based formation models. For her all these sub-classes are characterized by a comparable level of “interaction with an enabling digital technique rather than with an explicit representational structure as in the CAD model.” [Oxman, 2006:250] With respect to the above discussion this comparability is not justified. In the Turing-based model, therefore, the class of formation models got seperated into the two disjunct levels of operative and parametric awareness instead.

Norman Foster and Partners: Great Court Roof, British Museum, London, UK, 1999-2000
geometrically defined parametric model based on algebraic overlay of three surfaces; shape optimisation by method of relaxation

Oxman's third class of generative design models gets defined by the provision of complex computational mechanisms that deal with the emergence of forms that is this class is formed by methods of algorithmic morphogenesis. But only the sub-class of grammatical transformative design models is related to the process of form-making whereas the sub-class of evolutionary design models is not.

The difference between these two classes gets clearer if one looks at the Turing-model of computation. In that model, the morphogenetic process is nothing else than the transformation of the parameter $\mathbf{in}_{proj}$ into the visible form $\mathbf{out}_{proj}$ by means of an algorithmic description $\mathbf{T}_{proj}$. Furthermore, every algorithm $\mathbf{T}_{proj}$ generates a space **Var** of possible morphogenetic variations according to the range of the space of parameter **Para**. Evolutionary methods like genetic algorithms use this space of variation as search space. That is they are methods of optimisation that act as feedback-loop on the process of morphogenesis in order to scan the space **Var** for the best possible solution. (Figure 9) In other words, such methods are not part of the process of form-generation itself but are methods of evaluation of the produced form based on criteria of fitness.

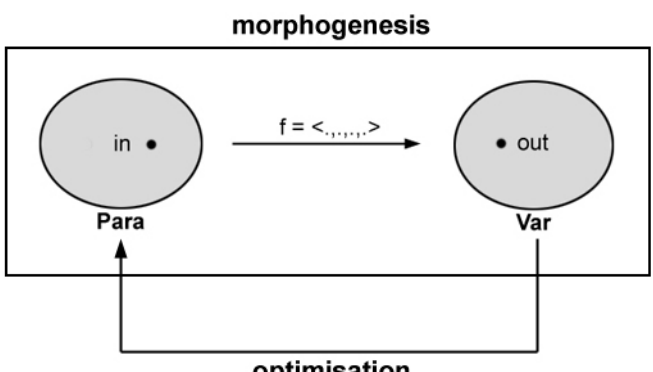
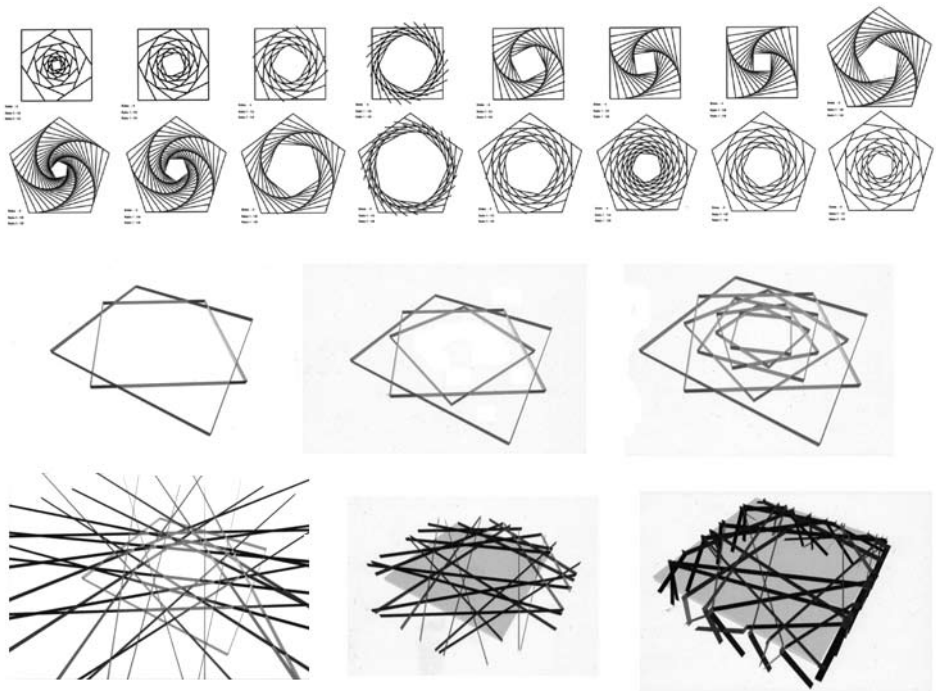


Figure 9: optimisation as external process



Toyo Ito: Serpentine Gallery Pavilion, London, UK, 2002
complex weave out of repeated nesting of rotated squares and extension into field of intersecting lines

Therefore, evolutionary methods should not be part of a class that tries to deal with the emergence of form but of a new class of methods of evaluation, instead. This new class should also contain the class of performance models because they act as methods of evalution outside of the morphogenetic process as well.

Conclusion

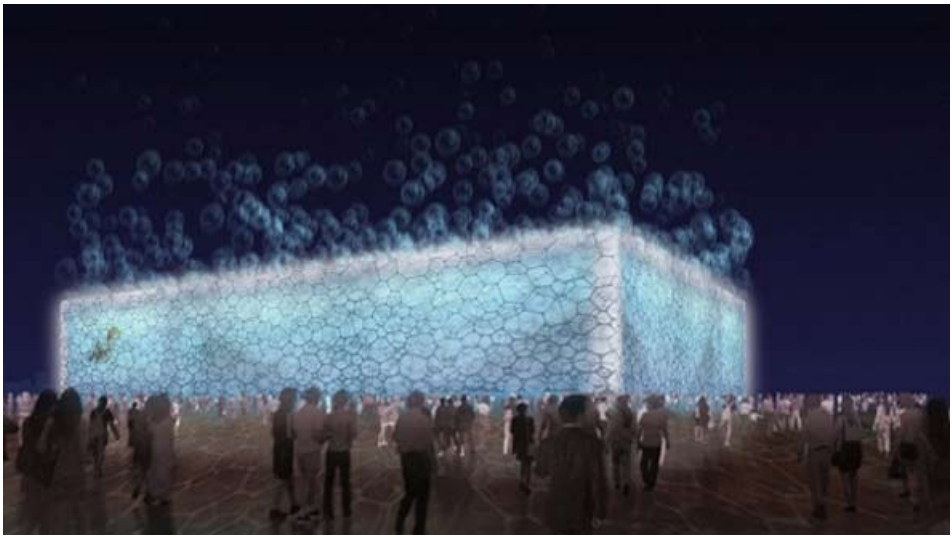
The discussion shows that the Turing-based model and the resulting division of the field of digital design into the operative, the parametric, and the algorithmic level can be seen as an attempt to separate the morphogenetic process in architecture from other computational methods. At the same time it establishes a framework of comparison for further investigations into formal methods of digital design and a possibility of integration of these methods into a theoretical discourse of architecture.

For instance, already this rough framework of algorithmic consciousnes illustrates that the equation of the digital in architectural design with the so-called architecture of blobs, an argument often used in the reflection of contemporary architecture, is not adequate because it would mean a limitation of the digital to the operative level.

Hence, an intensive and critical discussion of the computer and its influence onto the design process and architectural thinking is indispensable. On the one hand, the goal has to be an understanding of the new phenomena related to the digital and the identification of the potential of development of the discipline of architecture. [Hensel et al, 2006; Kubo & Ferré, 2004; Spuybroek, 2004; Rahim, 2006] On the other hand, the digital will allow a fresh look on themes that have been inherent to the architectural debate for a long time like the relation between design and its production. [Aish, 2005; Fritz, 2006; Sheil, 2005]

Toyo Ito: Serpentine Gallery Pavilion, London, UK, 2002
complex weave out of repeated nesting of rotated squares and extension into field of intersecting lines

The threshold between digital and representative processes of generation of architectural form, therefore, is not something that separates. Rather, the digital way of designing and exploring architecture should be perceived as a extended form of expression that is interwoven with the non-digital in manifold ways. A weave that waits to be discovered!



PTW: National Swimming Center, Beijing, China, 2003-06
space frame as combinatorial arrangement of three different nodes and four different members into effective sub-division of three dimensional space similar to fundamental arrangement of organic cells or formation of soap bubbles

Notes

1. Former research into the design process has favored the view onto architecture from within and has been centered on the analysis and formal modeling of behavioral, procedural and the cognitive activities of designing. [Cross, 1984; Lawson, 1997; Mitchell, 1990]

2. A possible coding of data is the interpretation of the finite sequence $a_0a_1\ldots a_n$ as digits in a binary representation $(.)_2$ of the number, that is $(a_0a_1\ldots a_n)_2 = a_0 2^0 + a_1 2^1 + \ldots + a_n 2^n$.

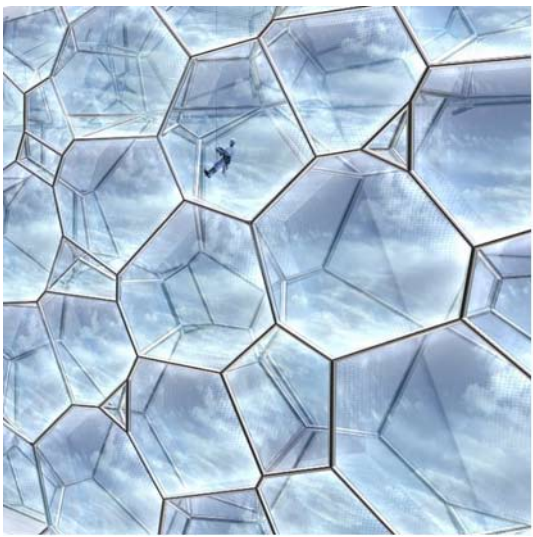
3. In order to work properly, every program p needs valid input data that is not every number **in** $\in \mathbb{N}$ is acceptable as argument. Therefore, p cannot be a total function in general.

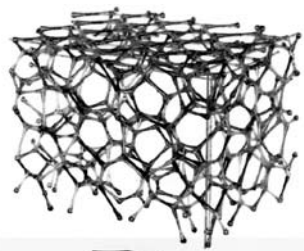
4. Hilbert believed that all mathematics could be precisely axiomatized. He thought that once this was done there would be an algorithm that would take as input any mathematical statement, and, after a finite number of steps, decide whether the statement was true or false. Restricted onto first-order logic this problem is known as the *Entscheidungsproblem*. It is this problem that gave impetus for the drive to codify the notion of computability.

5. This belief is known as the Curch-Turing Thesis and is uniformly accepted by mathematicians. For more details see [Copeland:2002]

6. The first recorded algorithm is that of Euclid for finding the greatest common divisor of two integers from around 300 BC. And the word ‘algorithm’ is derived from the name of the mathematician al-Khwarizmi, who worked at the court of Mamun in Baghdad around the early part of the 9^{th} century.

7. In architecture the computer gets applied not only in the form-generative process of design but in other context as well, e.g. process managment and collaboration (Building Information Modeling, BIM) or simulation of building performance. However, such applications are not in the focus of this examination. For more on this use of the computer in architetcure see [Kalay, 2004; Schodek, 2005; Steel, 2001]





PTW: National Swimming Center, Beijing, China, 2003-06
space frame as combinatorial arrangement of three different nodes and four different members into effective sub-division of three dimensional space similar to fundamental arrangement of organic cells or formation of soap bubbles

8. NURBS are nearly ubiquitous for computer-aided design (CAD), manufacturing (CAM), and engineering (CAE) and are part of numerous industry wide used standards, such as IGES, STEP, ACIS, and PHIGS.

9. The ability of most of the contemporary CAD-softwares to extend its functionality by plug-Ins, small programs defined by the user himself, makes clear that this point of view is not an unrealistic one.

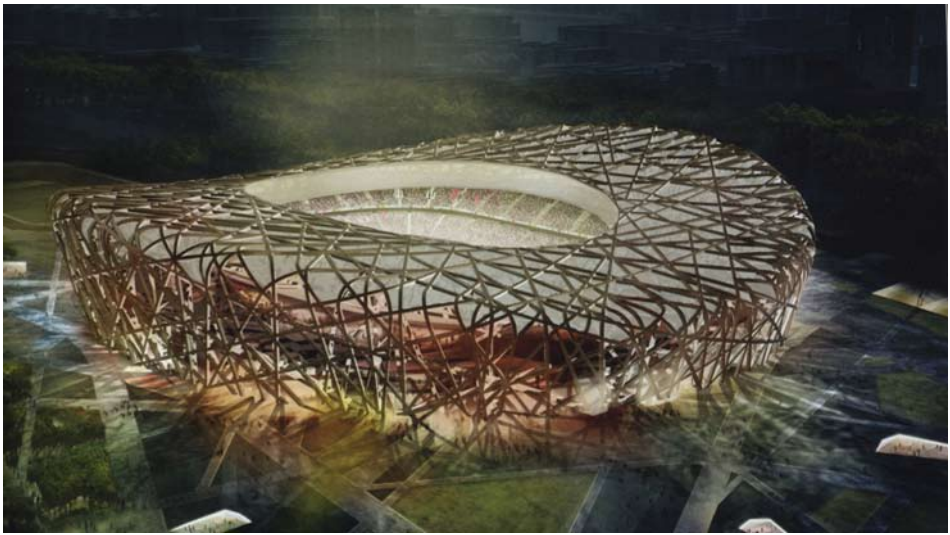
10. From a computational point of view, NURBS provide for an efficient data representation of geometric forms, using a minimum amount of data (control points, weights, degree) and relatively few steps for shape computation, which is why most of today's digital modeling programs rely on NURBS as a computational method for constructing complex surface models. [Kolarevic, 2003:15]

11. It is this shift from the operative to the time-based parametric that Greg Lynn pursues in his well-known book *Animate Form*. [Lynn, 1999]

12. See [De Luca & Nardini, 2002] for more details on these and further techniques.

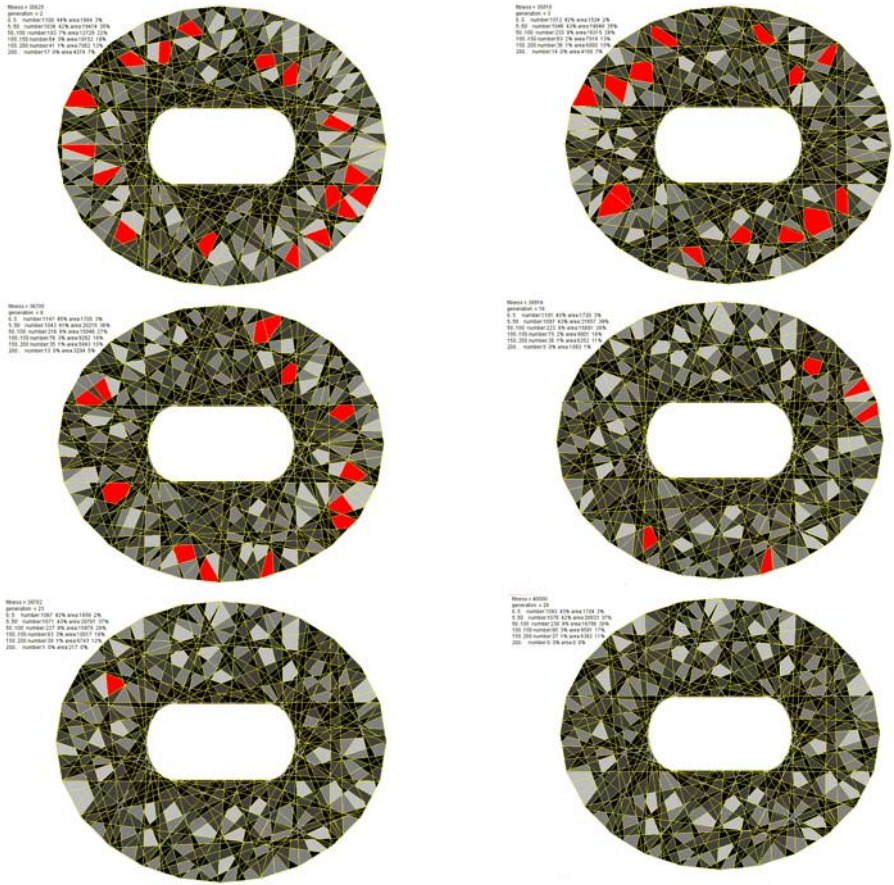
13. In general, a critical review of the potential of evolutionary algorithms for architectural design is indispensable. [Leach, 2006] Such methods lead often to a mapping of architectural quality onto an easy determinable numerical value and with it to a revival of a purely performative and functionalist view onto architecture. „Perhaps attention to performance will contribute to a new understanding of the ways buildings are imagined, made and experienced. But this new understanding will not result from the development and deployment of new techniques alone. The continued dedication to a technical interpretation of performance will lead to nothing more than a uncritical reaffirmation of old-style functionalist thinking – a kind of thinking that is both reductive and inadequate because it recognizes only what it can predict." [Leatherbarrow, 2004:7]

14. The integrated compound model is only the combination of the other models of digital design and can be excluded in this comparison.



Herzog & de Meuron: National Stadium, Beijing, China, 2002-07
surface of roof as complex spatial grid-like formation based on optimisation with respect of in-between space

15. An example for this limitation of digital architecture to the operative methode of topological deformation is Picon's statement that „until now, however, this debate for or against digital architecture has essentially focused on forms, on the value to be attributed to the 'blobs' and other 'folds' that one finds associated with signatures as different as Greg Lynn, UN Studio and Foreign Office." [Picon, 2004:59] With it he clearly refers to the journal Architectural Design and its special issue *Folding in Architecture*. This publication was edited by Greg Lynn and had an enormous influence onto the architectural debate in the 1990s. For a critical review of it see [Carpo, 2004]. A collection of build blob-architecture can be found in [Schmal, 2001].



Herzog & de Meuron: National Stadium, Beijing, China, 2002-07
surface of roof as complex spatial grid-like formation based on optimisation with respect of in-between space

Literature

Aish, Robert: *From Intuition to Precision,* AA Files, 52 (2005), 62-63

Balmond, Cecil: *Geometry, Algorithm, Pattern* in Leach, Neil / Turnball, David / Williams, Chris (ed.): *Digital Tectonics*, John Wiley, Chichester, USA, 2004

Barker-Plummer, David: *Turing Machines*, in Edward N. Zalta (ed.): *The Stanford Encyclopedia of Philosophy (Spring 2005 Edition)*, http://plato.stanford.edu/archives/spr2005/entries/turing-machine/

Carpo, Mario: *Ten years of folding*, in *Folding in architecture*, Revised Edition, John Wiley, Chichester, UK, 2004, 14-19

Cooper, S. Barry: *Computability theory*, Chapman & Hall/CRC, Boca Raton, USA, 2004

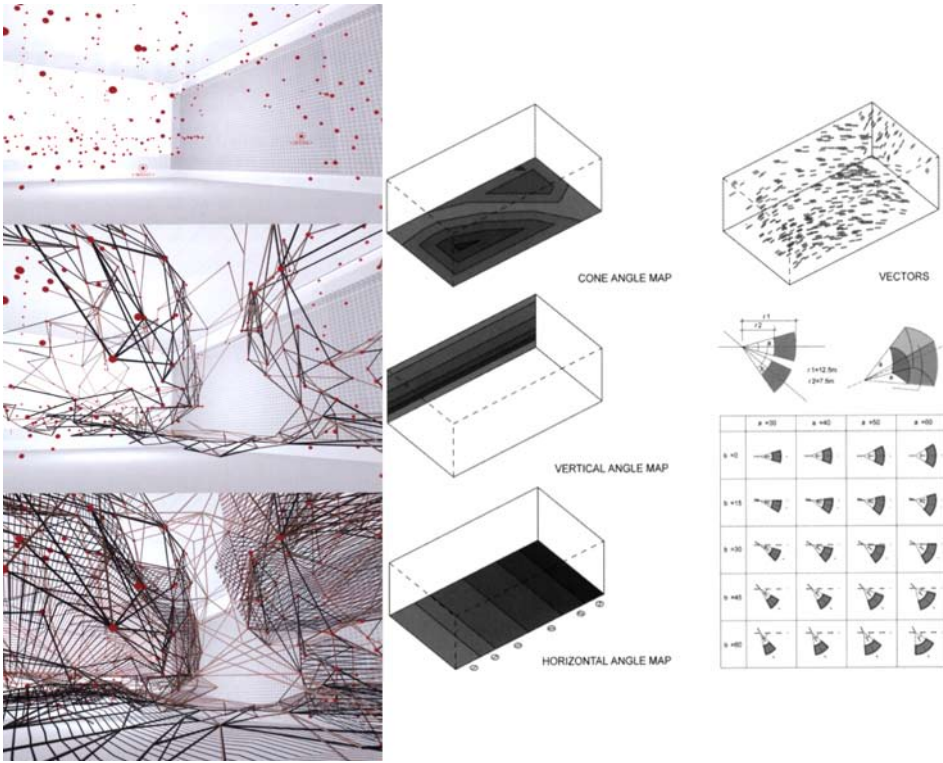
Copeland, B. Jack: The *Church-Turing Thesis*, in Edward N. Zalta (ed.): *The Stanford Encyclopedia of Philosophy (Fall 2002 Edition)*, http://plato.stanford.edu/archives/fall2002/entries/church-turing/

Cross, Nigel (ed.): *Developments in Design Methodology*, John Wiley, Chichester, UK, 1984

De Luca, Francesco & Nardini, Marco: *Avant-garde Techniques in Contemporary Design*, Birkhäuser, Basel, Switzerland, 2002

Fritz, Oliver: *Handwerk im Computerzeitalter*, Archithese, 4 (2006), 26-31

Hensel, Michael / Menges, Achim / Weinstock, Michael: *Techniques and Technologies in Morphogenetic Design*, Architectural Design, 76:2 (2006)



Ocean North: Music and Art Center, Jyväskylä, Finland, 2004-05 (design study, phase 2)
rule-based growth process of lattice system informed by performance requirements

Kalay, Yehuda E.: *Architecture's New Media – Principles, Theories, and Methods of Computer-Aided Design*, MIT Press, Cambridge, USA, 2004

Kolarevic, Branko (ed.): *Architecture in the digital age: design and manufacturing*, Taylor & Francis, Abingdon, USA, 2003

Kubo, Michael & Ferré, Albert (ed.): *Phylogenesis: foa's ark*, Actar, Barcelona, Spain, 2004

Lawson, Bryan: *How designers think*, Architectural Press, London, UK, 1997

Leach, Neil: *Digital Morphogenese,* Archithese, 4 (2006), 44-49

Leatherbarrow, David: *Architecture's unscripted performance*, in Kolarevic, Branko & Malkawi, Ali (ed.): *Performative Architecture: Beyond Instrumentality*, Taylor & Francis, Abingdon, UK, 2004, 6-19

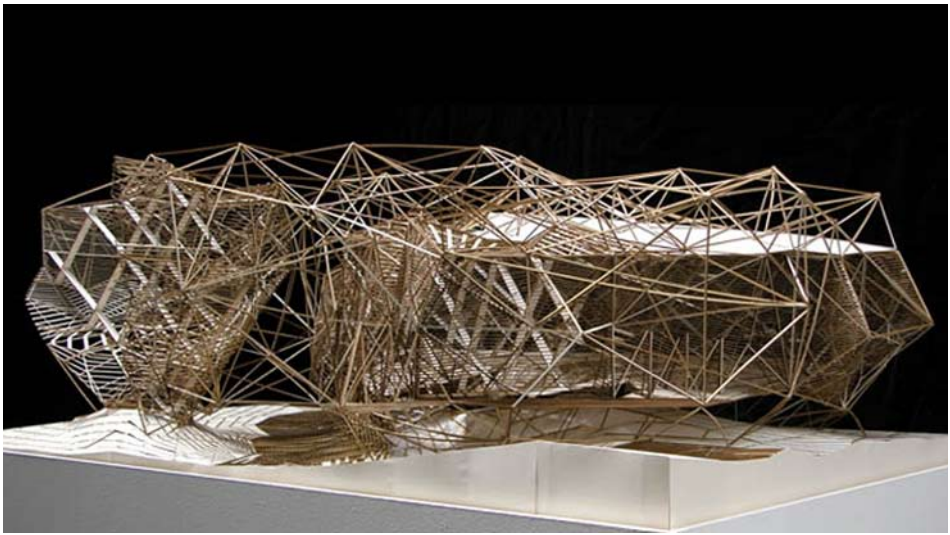
Lynn, Greg: *Animate Form*, Princeton Architectural Press, New York, USA, 1999

Mitchell, William J.: *The logic of architecture: design computation and cognition*, MIT Press, Cambridge, USA, 1990

Oxman, Rivka: *Theory and design in the first digital age*, Design Studies, 27(2006):229-265

Picon, Antoine: *Digital architecture and the poetics of computation*, in *Metamorph: Focus*, exhibition catalog, 9. International Architecture Exhibition, Venice, Italy, 2004, 58-69

Piegl, Les & Tiller, Wayne: *The NURBS Book. Monographs in Visual Communications*, Springer, Berlin, Germany, 2000



Ocean North: Music and Art Center, Jyväskylä, Finland, 2004-05 (design study, phase 2)
rule-based growth process of lattice system informed by performance requirements

Rahim, Ali: *Catalytic Formations – Architecture and Digital Design*, Taylor & Francis, Abingdon, UK, 2006

Schmal, Peter C.: *Blobmeister: erste gebaute Projekte*, Birkhäuser, Basel, Switzerland, 2001

Schodek, Daniel et al.: *Digital design and manufacturing: CAD/CAM technologies in architecture*, John Wiley, Hoboken, USA, 2005

Sheil, Bob: *Transgression from drawing to making*, arq, 1 (2005), 21-32

Silver, Mike: *Towards a Programming Culture in the Design Arts*, Architectural Design, 76:4, 5-11
Spuybroek, Lars: *NOX*, Thames & Hudson, New York, USA, 2004

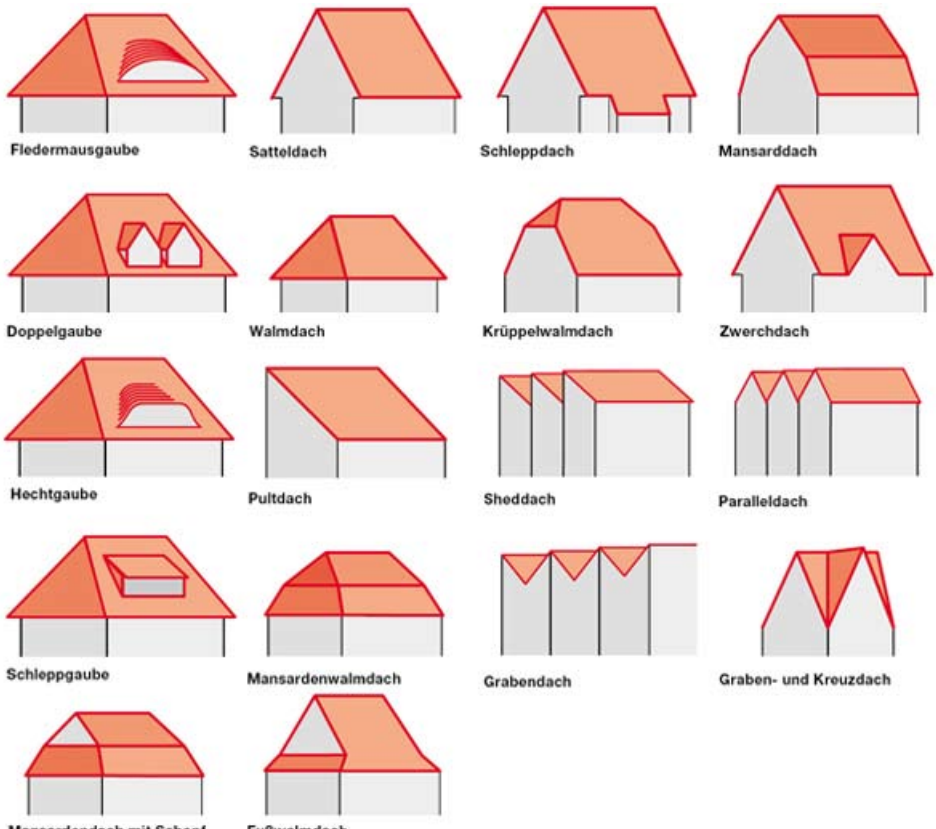
Steel, James: *Architecture and Computer: action and reaction in the digital revolution*, Laurence King, London, UK, 2001

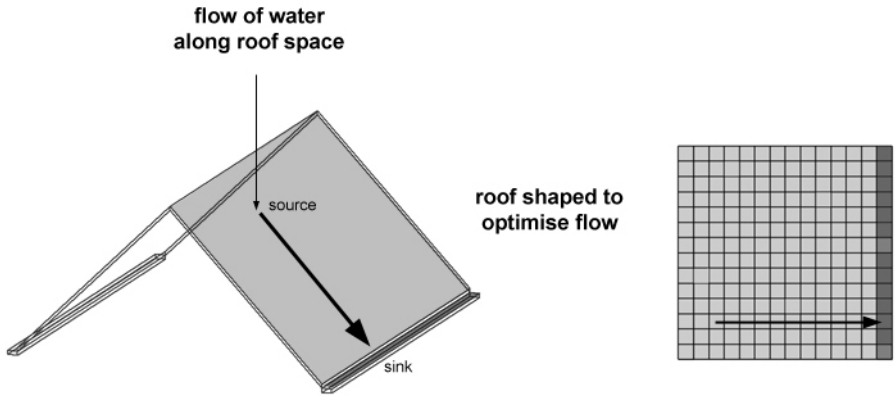
Szalapaj, Peter: *Contemporary Architecture and the Digital Design Process*, Architectural Press, Oxford, UK, 2005

Terzidis, Kostas: *Algorithmic Architecture*, Architectural Press, Oxford, UK, 2006

Turing, Alan: *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society, 2nd Series, 42(1936):230-265

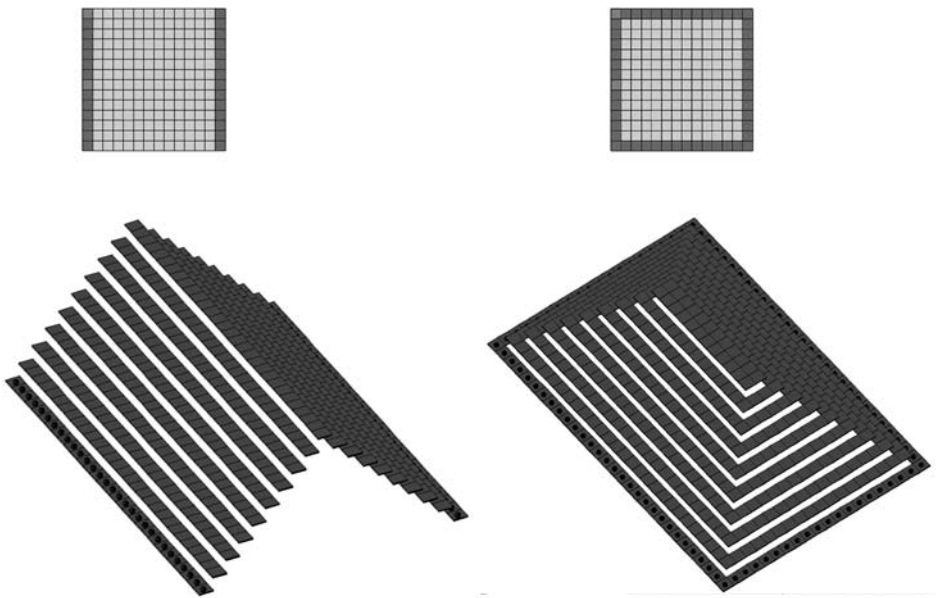
Roof Design


Fledermausgaube
Satteldach
Schleppdach
Mansarddach
Doppelgaube
Walmdach
Krüppelwalmdach
Zwerchdach
Hechtgaube
Pultdach
Sheddach
Paralleldach
Schleppgaube
Mansardenwalmdach
Grabendach
Graben- und Kreuzdach
Mansardendach mit Schopf
Fußwalmdach



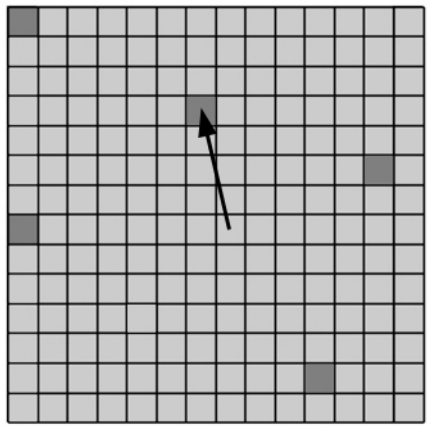
simple roof model:
placement of cells in height with respect to the
length of the shortest path from source to sink

The roof as an architectural element expresses a basic human need for protection against inclement environmental conditions. Over the centuries a number of typical roof types have been established with the resulting forms as efficient solutions to cover a building and protect its interior against wind and precipitation. Especially the ability of the roof for drainage has influenced this evolution of forms. The placement of the different sloped planes of the roof space can be seen as the visible trace of an optimised flow of water towards the drain. In other words, the roof can be modelled as a system of cells, the roof tiles, arranged in height. This spatial arrangement gets defined by the distance of the cell to the nearest drain. ...

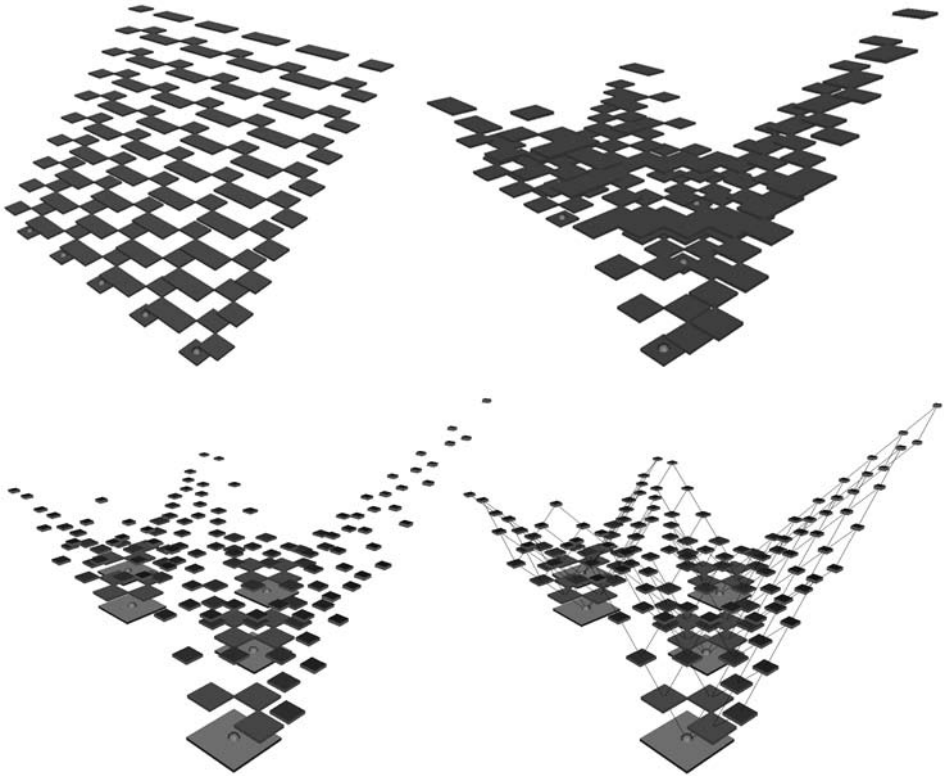


... It is this approach to the form of a roof that was be explored in more detail. That is the exploration of form according to a given set of rules. It enabled an extension of the well-known family of roof types. Already the above generation of a double pitched roof and a hipped roof showed that variations in this rule-based approach depend on the number of cells to drain and their location. ...

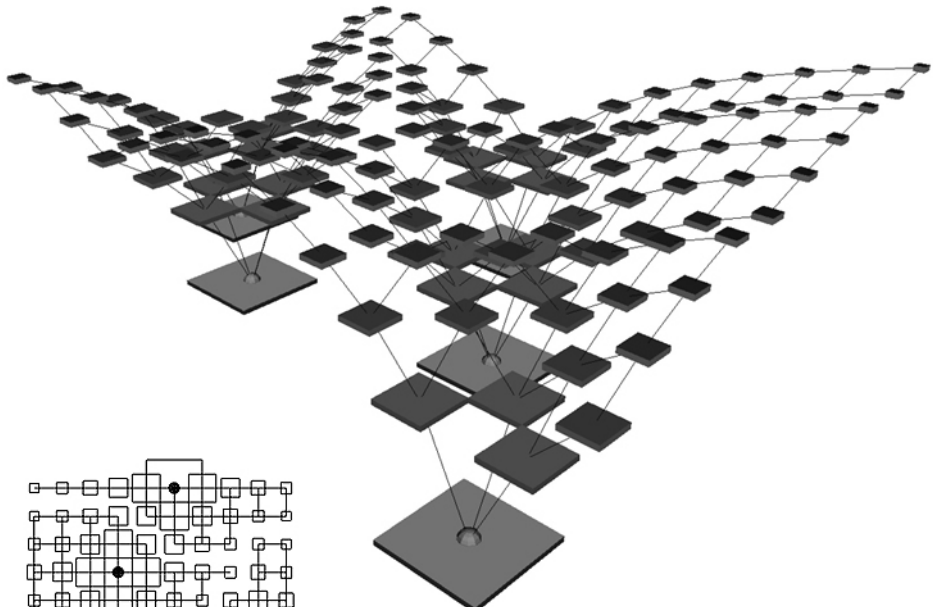
first generalization of roof model
location of sinks not restricted to border

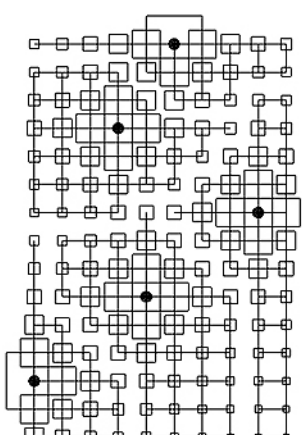


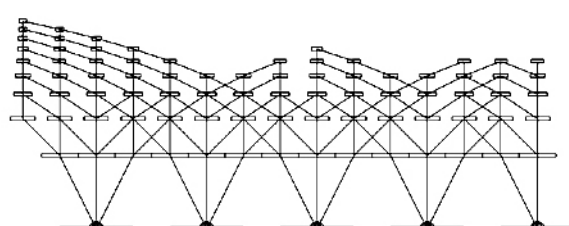
... This first step towards the design of a roof, therefore, generalized the existing typology in such a way that the number of sinks, i.e. cells to drain, and their placement in the cellular system could be varied. Already the reduction of sinks along the boundary changed the arrangement of the cells. However, it was the location of the sinks that had the most potential to influnce the development of form. ...

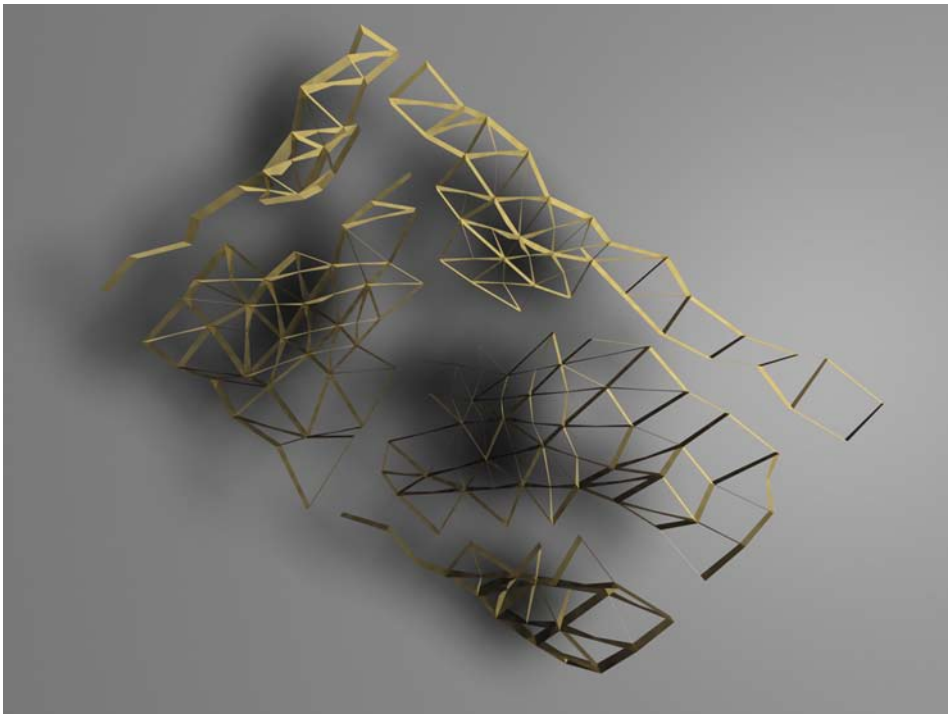


... In order to understand the development of form better the size of the cells were rescaled according to the distance of the cell to the nearest sink. Furthermore, the flow of water was made more visible by connecting cells respectively. In addition, the slope of the roof got rescaled by a logarithmic growth function. ...

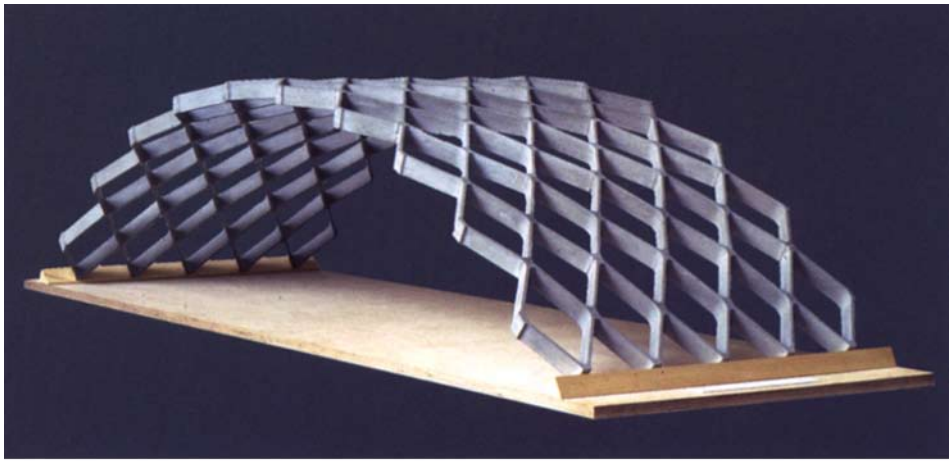


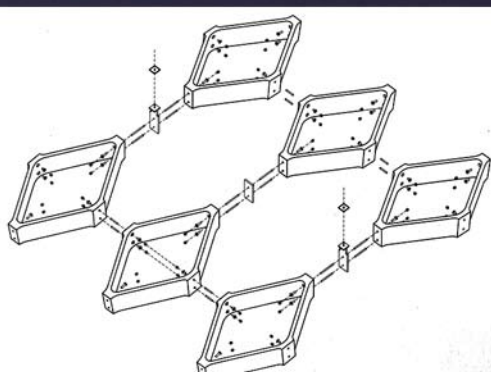


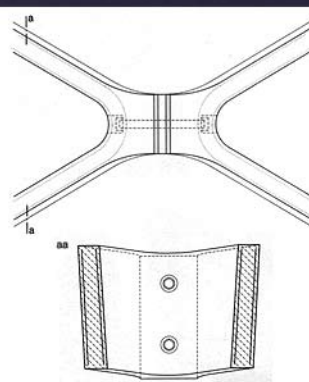




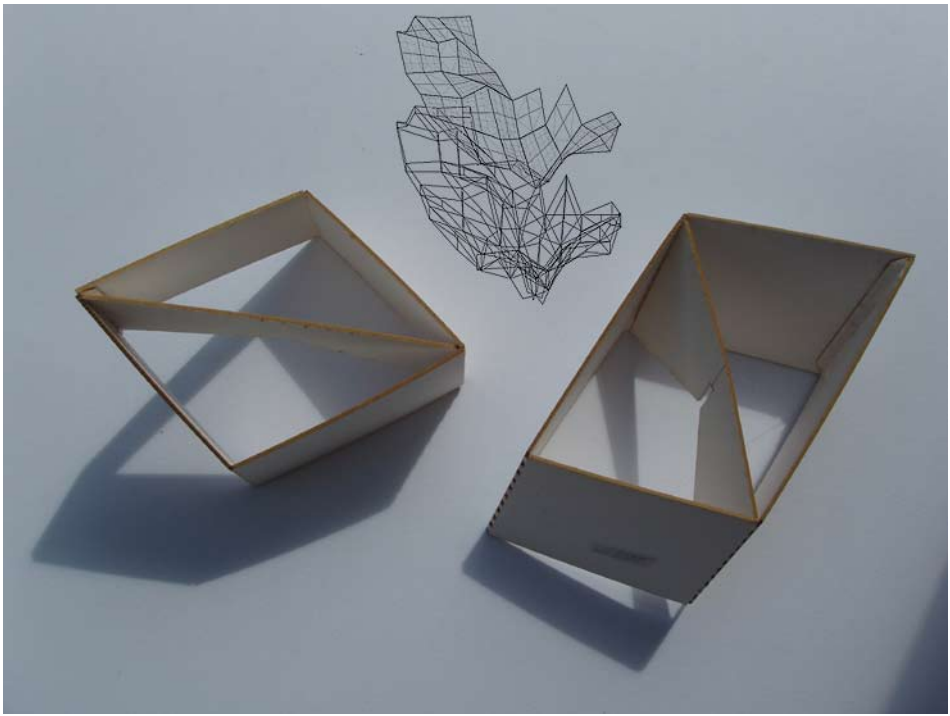
... The way cells were connected had a major influence on the further development because it led to a separation of the roof into components. To make this roof components buildable an adequate supporting structure was necessary. Therefore, a further exploration of the form was suspended in favour of physical experminents with the goal to find an appropriate structural systems. ...



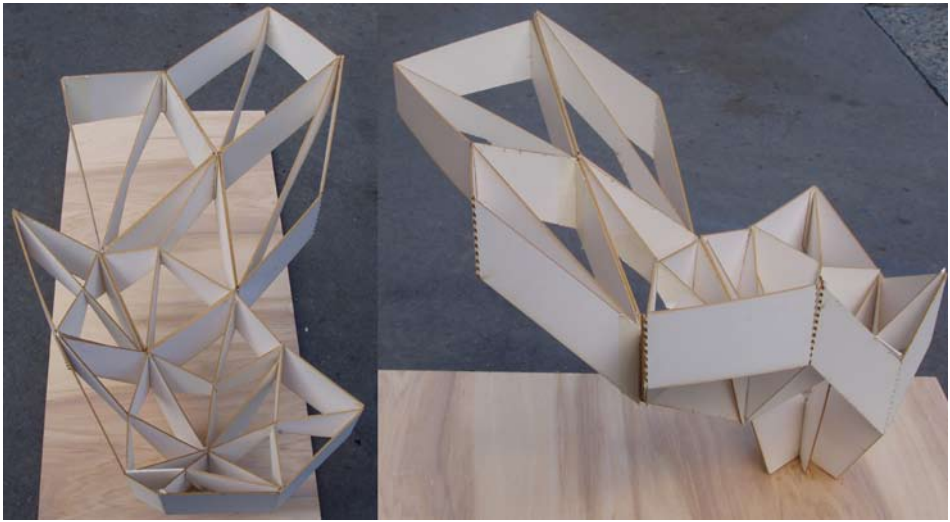




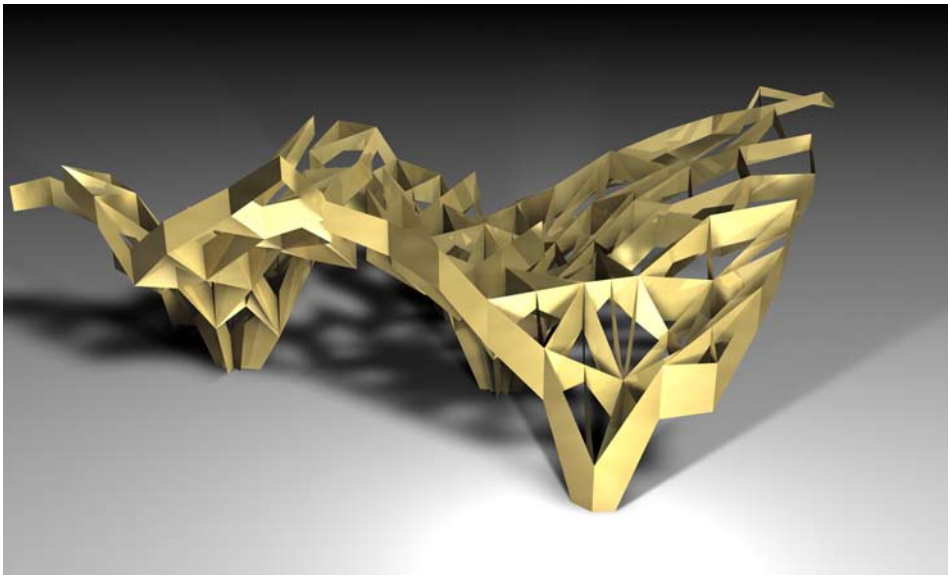
... Motivated by a diamond-lattice framing for arched forms developed by Friedrich Zollinger in 1905 an investigation into cellular structures started. ...



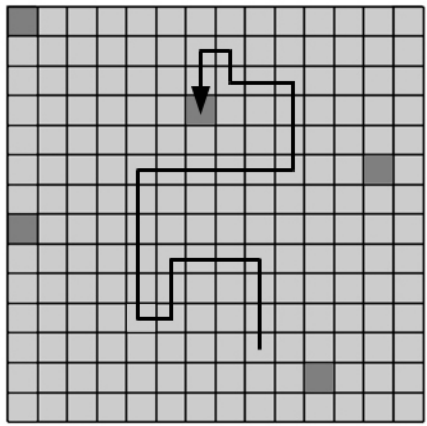
... This resulted in the development of stiffened cells with varying height according to their location in the final structure. The prefabricated cells were connected to each other leading to physical model of the supporting roof structure. ...



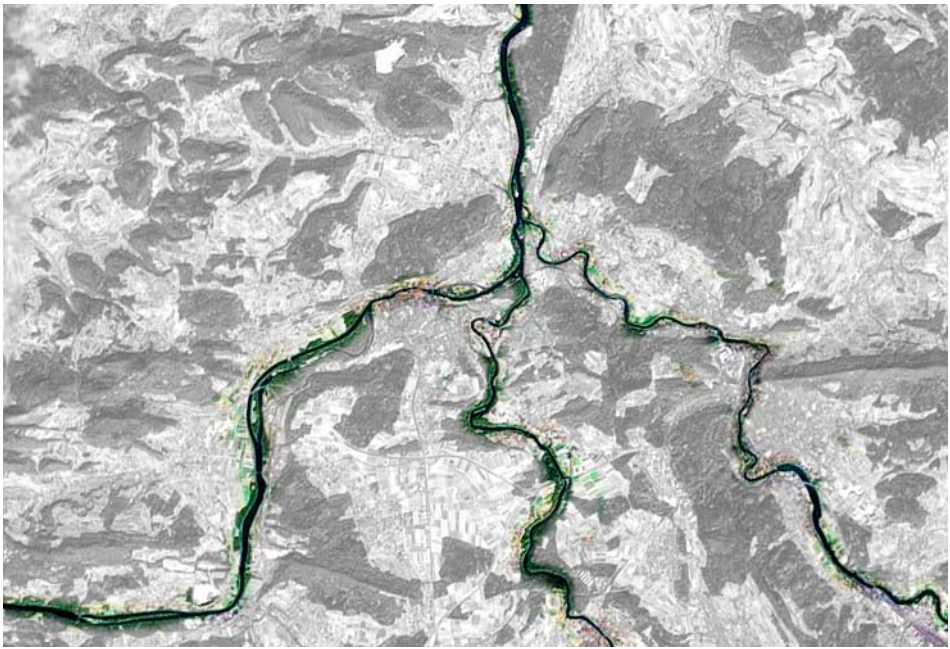
... This model proved very stable and the vaccuum forming of the roof surface increased the stability further. ...



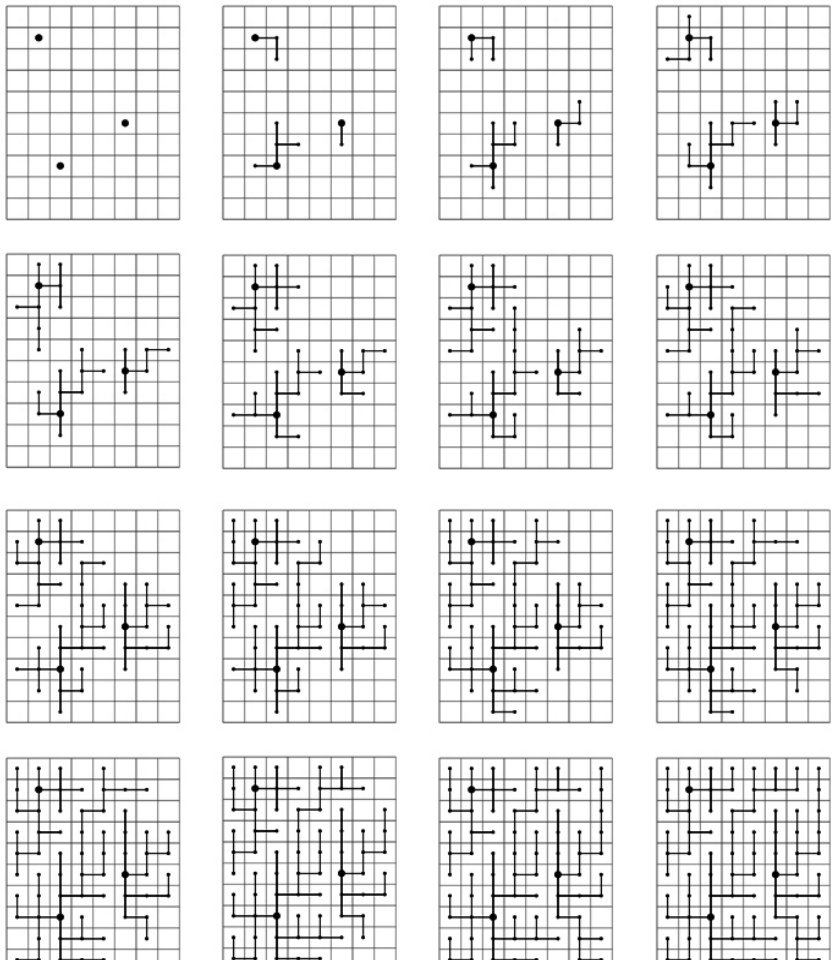
second generalization of roof model
flow not restricted to shortest path to sink

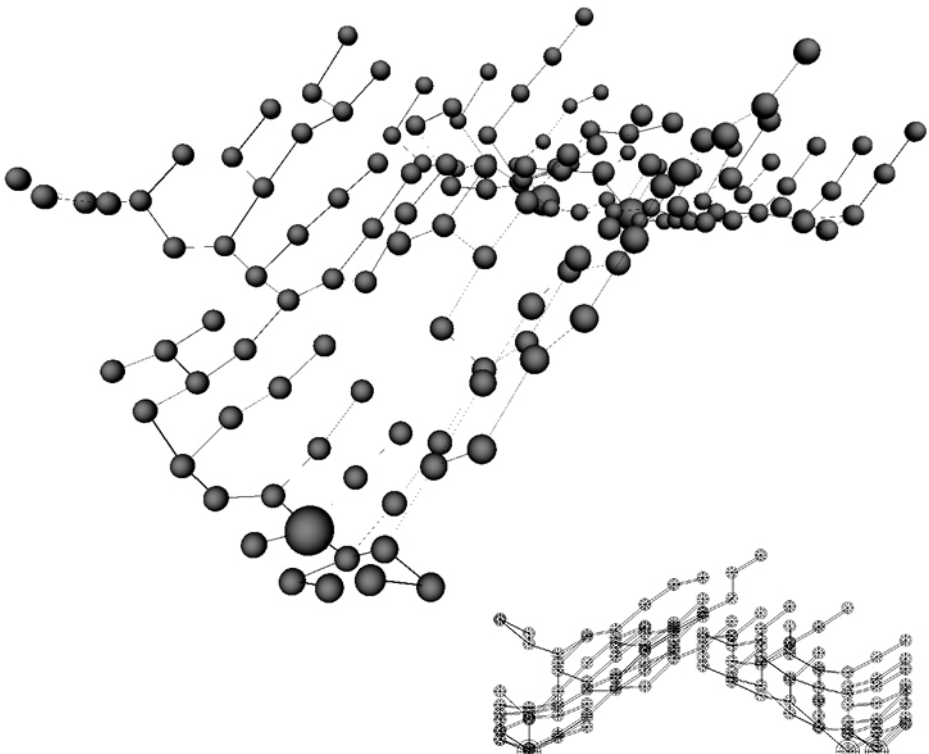


... Going back to the design process a further generalization of the system of rules was introduced. The restriction of the flow to the shortest path limited the space of variation. However, what is necessary for the functionality of the roof is not an optimised flow of water but the establishment of an arbitray way for the water to flow. ...



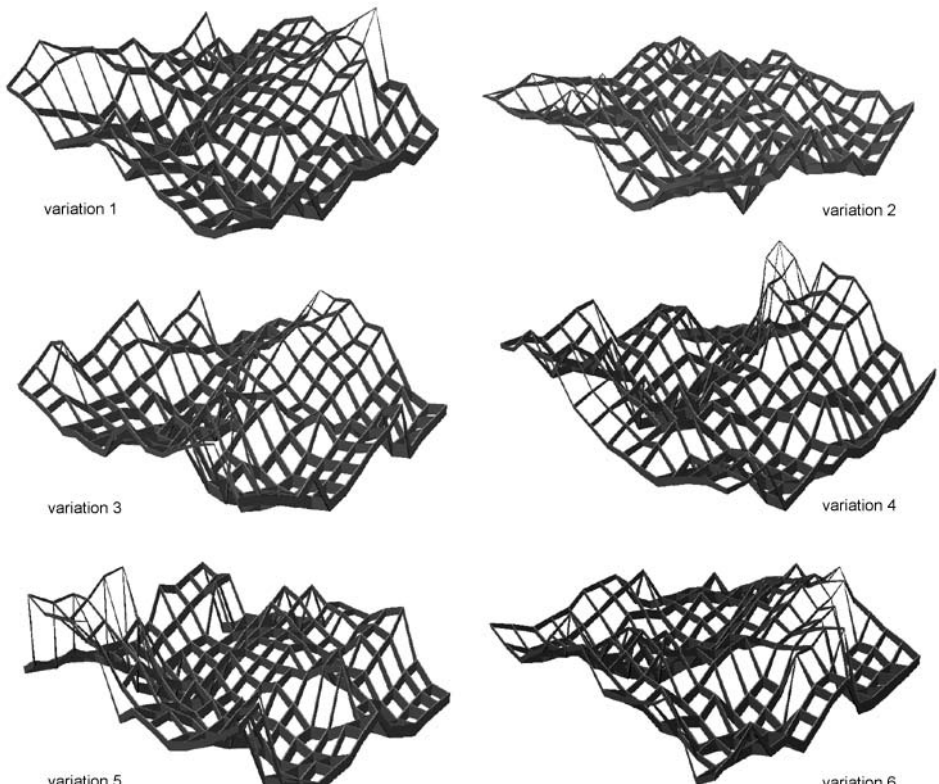
... Based on natural patterns of flow the rules were changed in such a way that a randomly generated system of rivers grew into the cellular structure that insured the drainage of water from every cell to one of the pre-defined sinks. ...



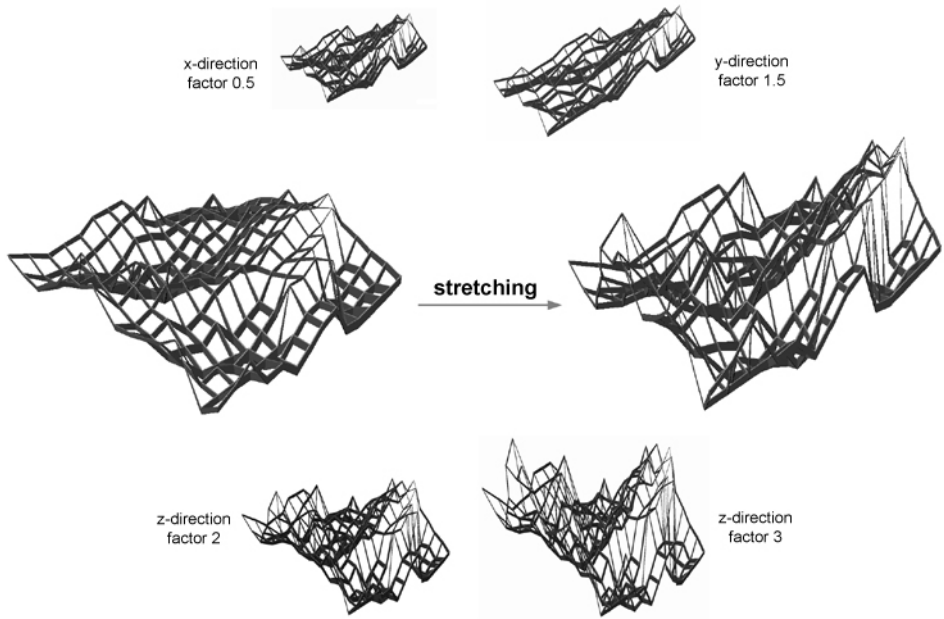


... Along this river system every cell received a position in space with respect to the distance to the sink it got attracted to which produced a logical notation for the form of the roof space. ...

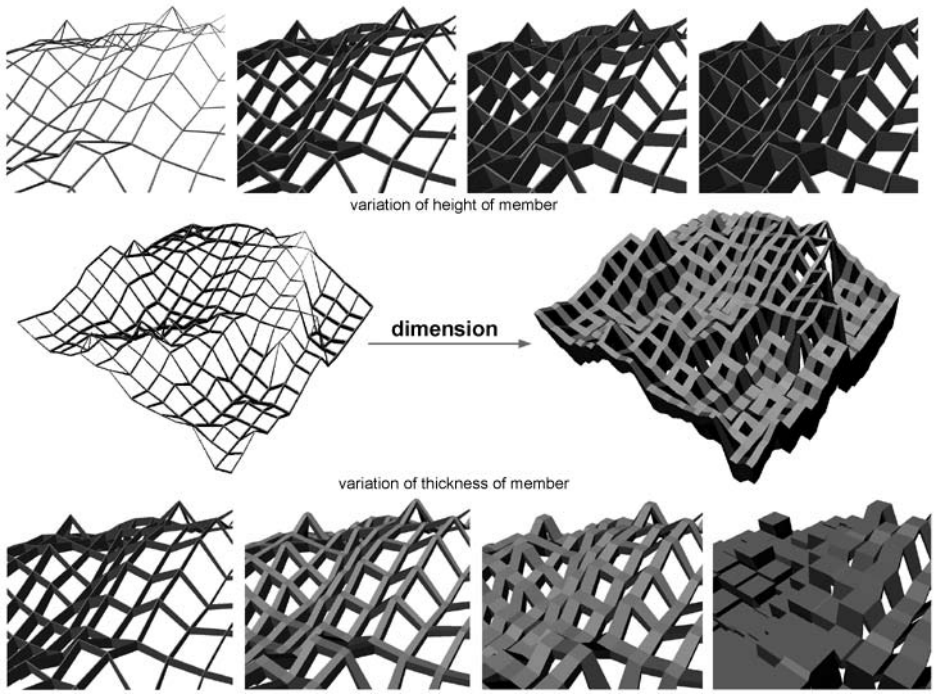
... The water cascade produced by the roof for a school yard in Zurich demonstrates the principle of the notation very well: Squared roof plates are directed and arranged in height in such a way that the water runs from roof to roof until it drops down. ...



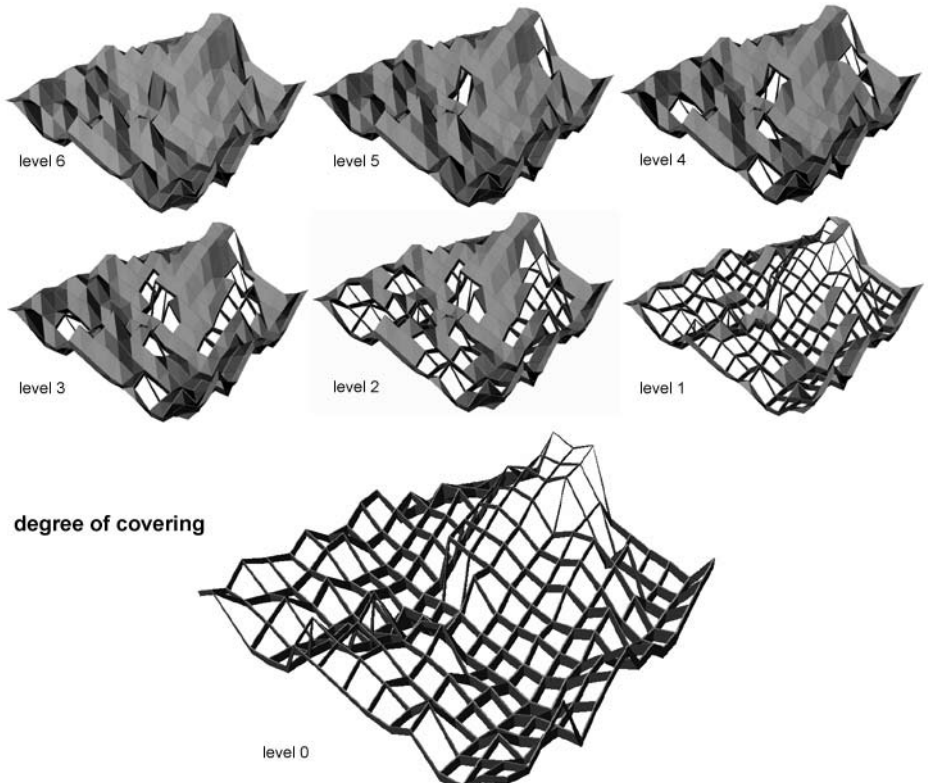
... Instead of dividing up the roof space into components the notation was used to deform the basic grid into a landscape formed by flow of water. This way a statically more stable roof structure could be achieved. ...



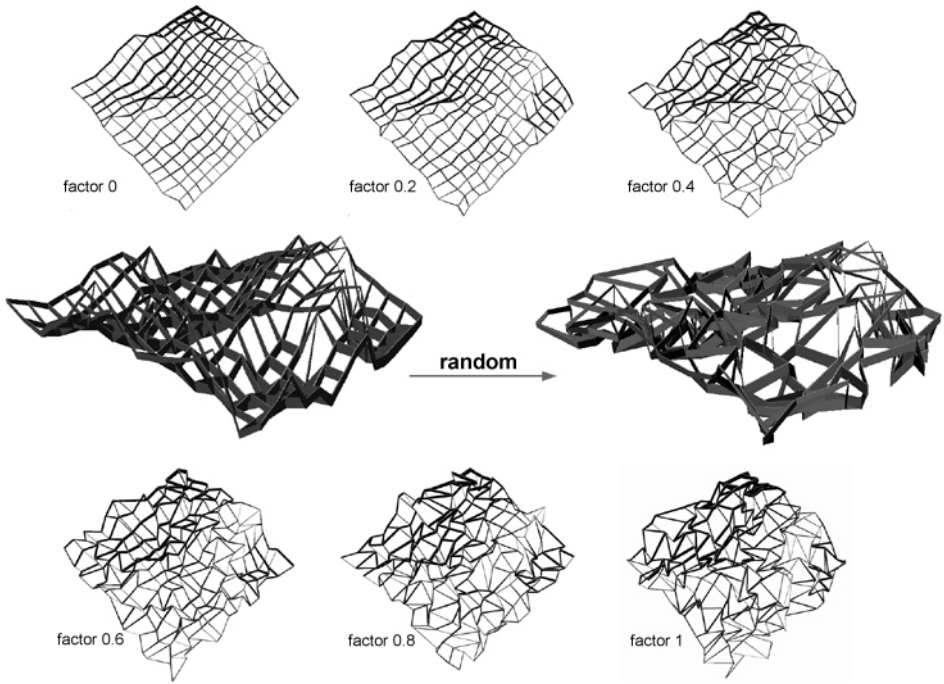
... For every of the produced variations a number of tools were programmed that allowed the deformation and adaptation of the roof surface. The first one was a stretching of the space in all three spatial directions. ...



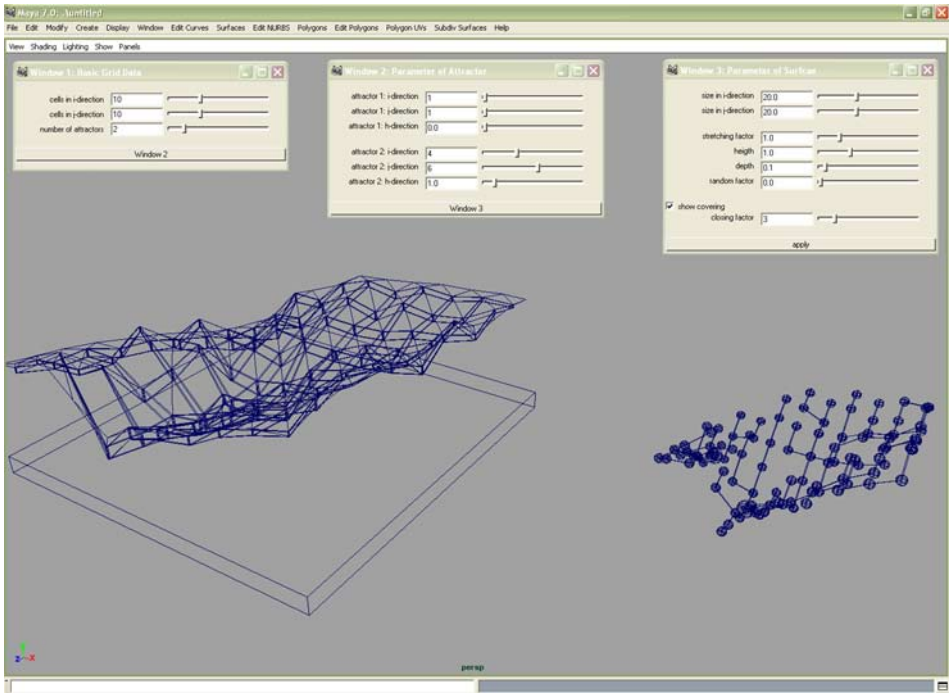
.... The second one was the ability to change the dimension of the structural grid that is the height of the elements and their thickness. ...



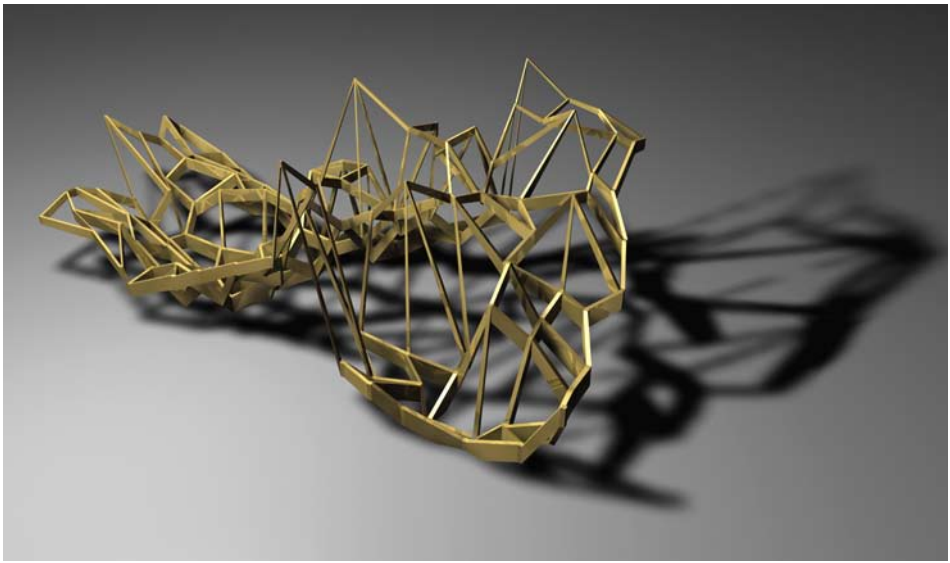
... A further tool was the degree of covering according to the stepness of the cell. This way a roof tiles could be changed into an opening. ...



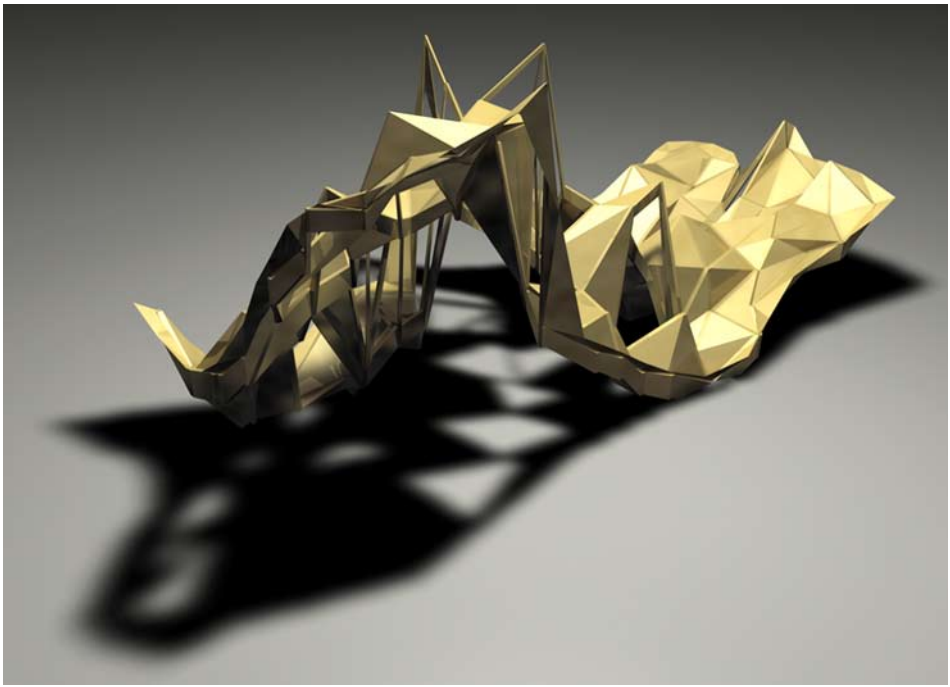
... Last but not least a random factor could be added to give the grid a more organic impression. ...



... All these tools were arranged in the third window. The other windows were used to define the grid size, the number of sinks and their location. In addition, the deformed roof surface and its supporting structures as well as the logical structure of the roof were displayed. ...

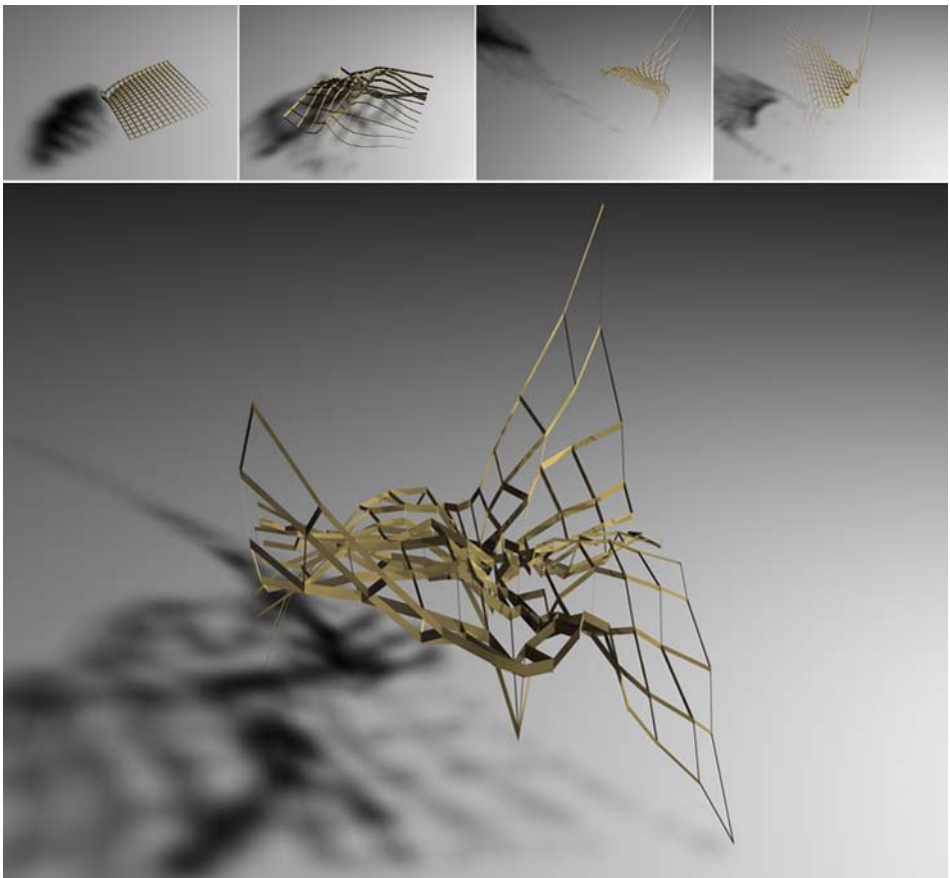


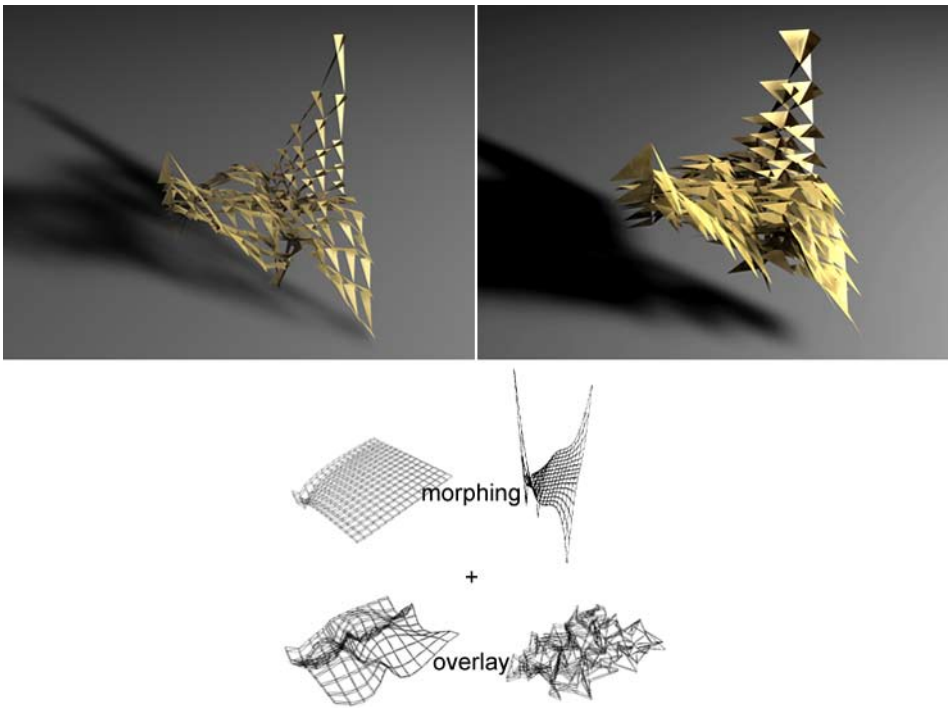
... Topview and supporting structure of an example based on a 15x10-grid with four sinks. ...



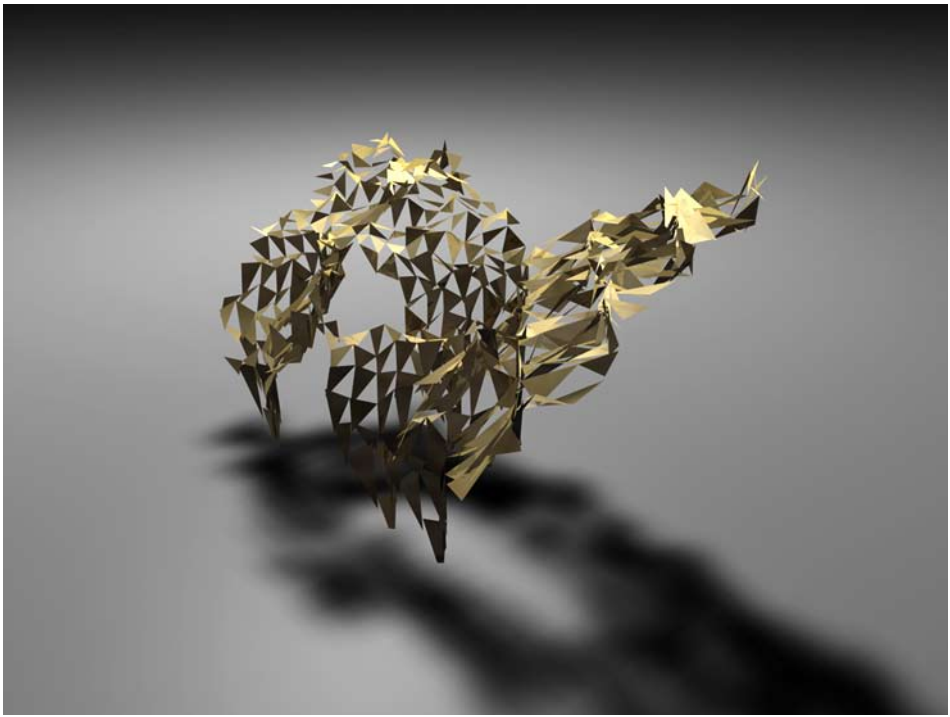
... This example makes clear that such a roof can easily be used for more than just as a covering of spaces. The roof itself produces spatial differentiation and varying degrees of openness and, therefore, has the potential to be considered as an architectural project. That is the process of generalization has extended the functionality of the roof and starts to dissolve the traditional boundary between classical elements of architectural producation, i.e. floor, wall, ceiling. ...

... This kind of formal exploration, the extracting and generalizing of rules of formation, by means of scripting shifts the process of design into the proximity of experiments common to the natural sciences. That is the digital opens up the possibility of incorporating scientific methods into the discipline of architecture. This helps to establish a non-metaphoric relation between science and architectural design and with it a new approach to architectural knowledge. ...

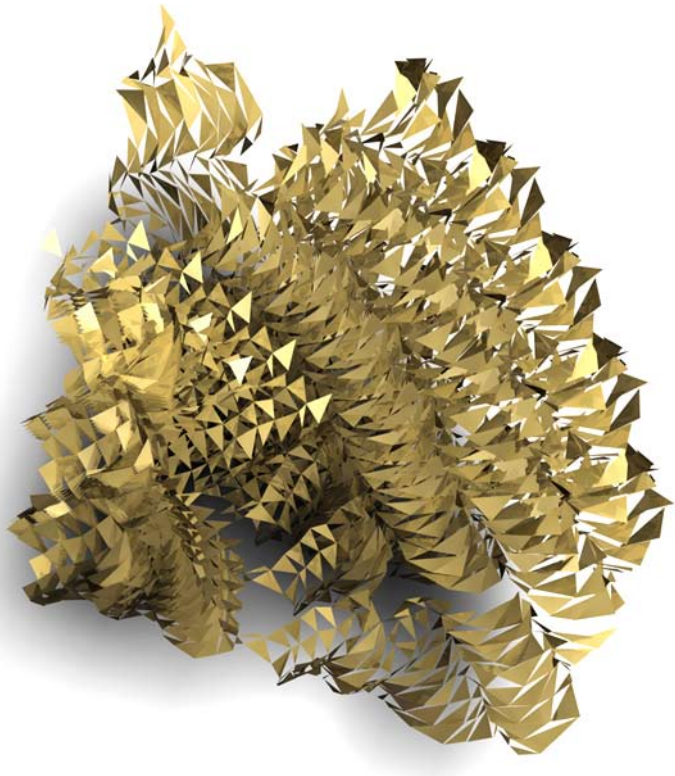




... A further line of design experimentation was established by deforming the basic arrangement of the grid by means of methods of morphing and overlay with other functions like trigonometric functions or hyperbolic ones. These functions are not compatible with the produced flow of water. Therefore, the type of covering was changed into leave-like sunshading. ...



... This led to a different expression of the roof pending between a perforated wall and a tree-like covering.



Appendix: MEL-Script

MEL-Script of Roof Design

```
global float $posX[];
global float $posY[];
global float $posZ[];
global int $attractor[];
global int $attractTo[];
global int $connectTo[];
global float $distance[];
global float $moveFlag[];
//-------------------------------------------------
// calculate row iPos and column jPos
// out of array-position n
//-------------------------------------------------
proc int jPos (int $n, int $m){
  int $ni = floor($n/$m);
  return $ni;
}
proc int iPos (int $n, int $m){
  int $ni = $n-floor($n/$m)*$m;
  return $ni;
}
//----------------------------------------
// calculate possible start point
// for new flow
//----------------------------------------
proc int CalcStartPoint(){
  global float $distance[];
  int $sPoint = 0;
  int $counter = 0;
  int $lengthArray = size($distance);
  if ($lengthArray > 0){
    float $availablePoint[];
    for ($i=0; $i<$lengthArray; $i++){
      if ($distance[$i] == -1){
```

```
      $availablePoint[$counter]=$i;
      $counter = $counter+1;
    }
   }
   $sPoint = floor(rand(0,$counter));
   $sPoint = $availablePoint[$sPoint];
  }
  return $sPoint;
}
//--------------------
// calculate path
// of new flow
//--------------------
proc CalcPath (int $sPoint, int $g_i){
  global float $distance[];
  global float $posY[];
  global int $attractTo[];
  global int $connectTo[];
  int $flag = 0;
  int $g_j = size($distance)/$g_i;
  int $s_i = iPos($sPoint,$g_i);
  int $s_j = jPos($sPoint,$g_i);
  if (($s_j < ($g_j-1)) && (($distance[$sPoint+$g_i] >= 0) && ($distance[$sPoint] < 0))){
   $distance[$sPoint] = $distance[$sPoint+$g_i]+1;
   $attractTo[$sPoint] = $attractTo[$sPoint+$g_i];
   $connectTo[$sPoint] = $sPoint+$g_i;
   $posY[$sPoint] = $posY[$sPoint+$g_i]+rand(0,1);
   $flag = 1;
  }
  if (($flag == 0) && (($s_i > 0) && (($distance[$sPoint-1] >= 0) && ($distance[$sPoint] < 0)))){
   $distance[$sPoint] = $distance[$sPoint-1]+1;
   $attractTo[$sPoint] = $attractTo[$sPoint-1];
   $connectTo[$sPoint] = $sPoint-1;
   $posY[$sPoint] = $posY[$sPoint-1]+rand(0,1);
   $flag = 1;
```

```
  }
  if (($flag == 0) && (($s_j > 0) && (($distance[$sPoint-$g_i] >= 0) && ($distance[$sPoint] < 0)))){
    $distance[$sPoint] = $distance[$sPoint-$g_i]+1;
    $attractTo[$sPoint] = $attractTo[$sPoint-$g_i];
    $connectTo[$sPoint] = $sPoint-$g_i;
    $posY[$sPoint] = $posY[$sPoint-$g_i]+rand(0,1);
    $flag = 1;
  }
  if (($flag == 0) && (($s_i < ($g_i-1)) && (($distance[$sPoint+1] >= 0) && ($distance[$sPoint] < 0)))){
    $distance[$sPoint] = $distance[$sPoint+1]+1;
    $attractTo[$sPoint] = $attractTo[$sPoint+1];
    $connectTo[$sPoint] = $sPoint+1;
    $posY[$sPoint] = $posY[$sPoint+1]+rand(0,1);
  }
}
//----------------------
// calculate height
//----------------------
proc float CalcMoveUp(float $x){
  float $xUp = 0;
  int $nx = floor($x);
  for ($k=0; $k<$nx; $k++){
    float $kMod = (float) $k;
    $xUp += (3/($kMod+1));
  }
  return $xUp;
}
//-------------------------------------------------
// open first window:
// input of data of grid
//-------------------------------------------------
if (`window -q -exists inputBasicGridData` == 1){
  deleteUI inputBasicGridData;
}
```

```
if (`window -q -exists inputParameterSurface` == 1){
  deleteUI inputParameterSurface;
}
if (`window -q -exists inputParameter` == 1){
  deleteUI inputParameter;
}
window -title «Window 1: Basic Grid Data» -wh 400 300 inputBasicGridData;
  columnLayout -adjustableColumn true;
  text «   «;
  intSliderGrp
    -label «cells in i-direction»
    -field true
    -minValue 1
    -maxValue 30
    -value 10
    iDirection;
  intSliderGrp
    -label «cells in j-direction»
    -field true
    -minValue 1
    -maxValue 30
    -value 10
    jDirection;
  intSliderGrp
    -label «number of attractors»
    -field true
    -minValue 1
    -maxValue 8
    -value 1
    noAttractor;
  text «  «;
  button -label «Window 2» -width 30 -height 20 -command inputParameter;
showWindow inputBasicGridData;
```

```
//--------------------------------------------------
// read grid data
//--------------------------------------------------
proc inputParameter(){
  int $grid_i = `intSliderGrp -q -value iDirection`;
  int $grid_j = `intSliderGrp -q -value jDirection`;
  int $numberAttractor = `intSliderGrp -q -value noAttractor`;
  //---------------------------------
  // open second window:
  // input of data of attractor
  //---------------------------------
  if (`window -q -exists inputParameter` == 1){
    deleteUI inputParameter;
  }
  window -title «Window 2: Parameter of Attractor» -wh 400 300 inputParameter;
    columnLayout -adjustableColumn true;
    text «   «;
    for ($k=1; $k<=$numberAttractor; $k++){
      string $nameX = «i_attractor» + $k;
      string $nameY = «h_attractor» + $k;
      string $nameZ = «j_attractor» + $k;
      string $textX = «attractor « + $k +»: i-direction»;
      string $textY = «attractor « + $k +»: h-direction»;
      string $textZ = «attractor « + $k +»: j-direction»;
      float $attHeight = (float) (($grid_i+$grid_j)/2);
      intSliderGrp
        -label $textX
        -field true
        -minValue 1
        -maxValue $grid_i
        -value 1
        $nameX;
      intSliderGrp
        -label $textZ
        -field true
```

```
        -minValue 1
        -maxValue $grid_j
        -value 1
        $nameZ;
      floatSliderGrp
        -label $textY
        -field true
        -minValue 0
        -maxValue $attHeight
        -value 0
        $nameY;
      text «   «;
    }
    button -label «Window 3» -width 30 -height 20 -command calcGrid;
  showWindow inputParameter;
}
//-----------------
// initialze grid
//-----------------
proc calcGrid(){
  int $grid_i = `intSliderGrp -q -value iDirection`;
  int $grid_j = `intSliderGrp -q -value jDirection`;
  int $numberAttractor = `intSliderGrp -q -value noAttractor`;
  global float $posX[];
  for ($n=0; $n<($grid_i*$grid_j); $n++){
    $posX[$n] = 0;
  }
  global float $posY[];
  for ($n=0; $n<($grid_i*$grid_j); $n++){
    $posY[$n] = -1;
  }
  global float $posZ[];
  for ($n=0; $n<($grid_i*$grid_j); $n++){
    $posZ[$n] = 0;
  }
```

```
global float $distance[];
for ($n=0; $n<($grid_i*$grid_j); $n++){
  $distance[$n] = -1;
}
global float $moveFlag[];
for ($n=0; $n<($grid_i*$grid_j); $n++){
  $moveFlag[$n] = 1;
}
global int $attractTo[];
for ($n=0; $n<($grid_i*$grid_j); $n++){
  $attractTo[$n] = 0;
}
global int $connectTo[];
for ($n=0; $n<($grid_i*$grid_j); $n++){
  $connectTo[$n] = 0;
}
int $index = 0;
global int $attractor[];
for ($i=0; $i<(2*$numberAttractor); $i){
  int $k = floor($i/2)+1;
  string $nameAttX = «i_attractor» + $k;
  string $nameAttY = «h_attractor» + $k;
  string $nameAttZ = «j_attractor» + $k;
  $attractor[$i]=`intSliderGrp -q -value $nameAttX`-1;
  $attractor[$i+1]=`intSliderGrp -q -value $nameAttZ`-1;
  $index = $attractor[$i]+$attractor[$i+1]*$grid_i;
  $attractTo[$index]=$index;
  $connectTo[$index]=$index;
  $distance[$index] = 0;
  $moveFlag[$index] = 0;
  $posY[$index] = `floatSliderGrp -q -value $nameAttY`;
  $i = $i+2;
}
int $loopNumber = 20*$grid_i*$grid_j;
```

```
  for ($i=0; $i<$loopNumber; $i++){
    int $startPoint;
    $startPoint = CalcStartPoint();
    CalcPath($startPoint,$grid_i);
  }
  paraSurface;
}
//-------------------------
// open third window:
// input of parameter
//-------------------------
proc paraSurface (){
  if (`window -q -exists inputParameterSurface` == 1){
    deleteUI inputParameterSurface;
  }
  window -title «Window 3: Parameter of Surfcae» -wh 400 300 inputParameterSurface;
    columnLayout -adjustableColumn true;
    text «   «;
    floatSliderGrp
      -label «size in i-direction»
      -field true
      -minValue 2
      -maxValue 50
      -value 20
      p_sizeX;
    floatSliderGrp
      -label «size in j-direction»
      -field true
      -minValue 2
      -maxValue 50
      -value 20
      p_sizeZ;
    text «   «;
    floatSliderGrp
      -label «stretching factor»
```

```
  -field true
  -minValue 0
  -maxValue 5
  -value 1
  p_s;
floatSliderGrp
  -label «heigth»
  -field true
  -minValue 0.1
  -maxValue 3
  -value 1
  p_t;
floatSliderGrp
  -label «depth»
  -field true
  -minValue 0
  -maxValue 2
  -value 0.1
  p_d;
floatSliderGrp
  -label «random factor»
  -field true
  -minValue 0
  -maxValue 1
  -value 0
  p_distort;
text «   «;
checkBox
  -label «show covering»
  -value true
  -width 80
  -align «left»
  p_cov;
intSliderGrp
  -label «closing factor»
```

```
      -field true
      -minValue 0
      -maxValue 20
      -value 3
      p_c;
    text «   «;
    button -label «apply» -width 30 -height 20 -command showGrid;
  showWindow inputParameterSurface;
}
//-------------------
// output of grid
//-------------------
proc showGrid (){
  select -all;
  delete;
  global float $posX[];
  global float $posY[];
  global float $posZ[];
  global int $attractor[];
  global int $attractTo[];
  global int $connectTo[];
  global float $distance[];
  global float $moveFlag[];
  float $posXCopy[];
  float $posYCopy[];
  float $posZCopy[];
  float $posYShadow[];
  float $shift[];
  int $grid_i = `intSliderGrp -q -value iDirection`;
  int $grid_j = `intSliderGrp -q -value jDirection`;
  int $numberAttractor = `intSliderGrp -q -value noAttractor`;
  float $sizeX = `floatSliderGrp -q -value p_sizeX`;
  float $sizeZ = `floatSliderGrp -q -value p_sizeZ`;
  float $relX = $sizeX/(float) $grid_i;
  float $relZ = $sizeZ/(float) $grid_j;
```

```
float $stretching = `floatSliderGrp -q -value p_s`;
float $closing = `intSliderGrp -q -value p_c`;
float $distort = `floatSliderGrp -q -value p_distort`;
float $thickness = `floatSliderGrp -q -value p_t`;
float $gSize = `floatSliderGrp -q -value p_d`;
float $gShift = $gSize/2;
for ($n=0; $n<($grid_i*$grid_j); $n++){
  $posYShadow[$n] = $posY[$n];
}
float $maxH = 0;
float $maxD = 0;
for ($i=0; $i<size($posY); $i++){
  if ($posY[$i]>$maxH){
    $maxH = $posY[$i];
  }
  if ($distance[$i]>$maxD){
    $maxD = (float) $distance[$i];
  }
}
float $relH = $thickness/($maxD+1);
float $i_pos;
float $j_pos;
for ($i=0; $i<size($distance); $i++){
  $i_pos = (float) iPos($i,$grid_i);
  $j_pos = (float) jPos($i,$grid_i);
  $posX[$i] = $i_pos;
  $posZ[$i] = $j_pos;
  if ($moveFlag[$i] == 1){
    $posXCopy[$i] = $posX[$i]*$relX+$distort*rand(-1,1)*$relX;
    $posZCopy[$i] = $posZ[$i]*$relZ+$distort*rand(-1,1)*$relZ;
  } else {
    $posXCopy[$i] = $posX[$i]*$relX;
    $posZCopy[$i] = $posZ[$i]*$relZ;
  }
  float $distortHeight = 0;
```

```
  if ($distance[$i] > 0){
    float $lowDistort = -1/$distance[$i];
    float $heighDistort = 1/($distance[$i]+1);
    $distortHeight = $distort*rand($lowDistort,$heighDistort);
  }
  float $relHeight = $posY[$i] - $posY[$attractTo[$i]];
  $posYCopy[$i] = $posY[$attractTo[$i]]+($relHeight+$distortHeight)*$stretching;
  float $relDistance = (float) $distance[$i]*$relH;
  $posYShadow[$i] = $posYCopy[$i]-($thickness-$relDistance);
  $shift[$i]=$gShift-3*$distance[$i]*$gShift/(4*$maxD);
}
// ------------
for ($i=0; $i<size($distance); $i++){
  float $cubeHeigth = $posYCopy[$i]-$posYShadow[$i];
  polyCube -width (2*$shift[$i]) -height $cubeHeigth -depth (2*$shift[$i]);
  move -r $posXCopy[$i] ($posYCopy[$i]-$cubeHeigth/2) $posZCopy[$i];
}
for ($i=0; $i<size($distance); $i++){
  int $g_i = iPos($i,$grid_i);
  int $g_j = jPos($i,$grid_i);
  if ($g_i < ($grid_i-1)){
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]-$shift[$i])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYCopy[$i+1] ($posZCopy[$i+1]-$shift[$i+1])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYShadow[$i+1] ($posZCopy[$i+1]-$shift[$i+1])
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]-$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYCopy[$i+1] ($posZCopy[$i+1]+$shift[$i+1])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYShadow[$i+1] ($posZCopy[$i+1]+$shift[$i+1])
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]-$shift[$i])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYCopy[$i+1] ($posZCopy[$i+1]-$shift[$i+1])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYCopy[$i+1] ($posZCopy[$i+1]+$shift[$i+1])
```

```
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]-$shift[$i])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYShadow[$i+1] ($posZCopy[$i+1]-$shift[$i+1])
    -p ($posXCopy[$i+1]-$shift[$i+1]) $posYShadow[$i+1] ($posZCopy[$i+1]+$shift[$i+1])
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i]) ;
  select -clear;
 }
 if ($g_j < ($grid_j-1)){
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i])
    -p ($posXCopy[$i+$grid_i]+$shift[$i+$grid_i]) $posYCopy[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i+$grid_i]+$shift[$i+$grid_i]) $posYShadow[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]-$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i])
    -p ($posXCopy[$i+$grid_i]-$shift[$i+$grid_i]) $posYCopy[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i+$grid_i]-$shift[$i+$grid_i]) $posYShadow[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i]-$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i])
    -p ($posXCopy[$i+$grid_i]+$shift[$i+$grid_i]) $posYCopy[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i+$grid_i]-$shift[$i+$grid_i]) $posYCopy[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i]-$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i]);
  polyCreateFacet
    -p ($posXCopy[$i]+$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i])
    -p ($posXCopy[$i+$grid_i]+$shift[$i+$grid_i]) $posYShadow[$i+$grid_i] ($posZCopy[$i+$grid_i]-
          $shift[$i+$grid_i])
    -p ($posXCopy[$i+$grid_i]-$shift[$i+$grid_i]) $posYShadow[$i+$grid_i] ($posZCopy[$i+$grid_i]-
```

```
            $shift[$i+$grid_i])
    -p ($posXCopy[$i]-$shift[$i]) $posYShadow[$i] ($posZCopy[$i]+$shift[$i]) ;
  select -clear;
}
// ------------
if (`checkBox -q -value p_cov` == 1){
  if (($g_i < ($grid_i-1)) && ($g_j < ($grid_j-1))){
    int $d1 = abs($distance[$i]-$distance[$i+1]);
    int $d2 = abs($distance[$i]-$distance[$i+$grid_i]);
    int $d3 = abs($distance[$i]-$distance[$i+$grid_i+1]);
    int $flowDiff = max($d1,$d2);
    $flowDiff = max($flowDiff,$d3);
    if ($flowDiff <= $closing){
      polyCreateFacet
        -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i])
        -p ($posXCopy[$i+1]-$shift[$i+1]) $posYCopy[$i+1] ($posZCopy[$i+1]+$shift[$i+1])
        -p ($posXCopy[$i+$grid_i+1]-$shift[$i+$grid_i+1]) $posYCopy[$i+$grid_i+1]
                        ($posZCopy[$i+$grid_i+1]-$shift[$i+$grid_i+1]);
      polyCreateFacet
        -p ($posXCopy[$i+$grid_i+1]-$shift[$i+$grid_i+1]) $posYCopy[$i+$grid_i+1]
                        ($posZCopy[$i+$grid_i+1]-$shift[$i+$grid_i+1])
        -p ($posXCopy[$i+$grid_i]+$shift[$i+$grid_i]) $posYCopy[$i+$grid_i]
                        ($posZCopy[$i+$grid_i]-$shift[$i+$grid_i])
        -p ($posXCopy[$i]+$shift[$i]) $posYCopy[$i] ($posZCopy[$i]+$shift[$i]);
      select -clear;
    }
  }
  float $bX = 0;
  float $bY = 0;
  float $bZ = 0;
  float $bXnext = 0;
  float $bYnext = 0;
  float $bZnext = 0;
  if (($g_j == 0) && ($g_i < ($grid_i-1))){
    $bX = ($posXCopy[$i]+$posXCopy[$i+1])/2;
```

```
    $bY = ($posYCopy[$i]+$posYCopy[$i+1])/2+0.25;
    $bZ = ($posZCopy[$i]+$posZCopy[$i+1])/2-0.5;
    polyCreateFacet
      -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
      -p $posXCopy[$i+1] $posYCopy[$i+1] $posZCopy[$i+1]
      -p $bX $bY $bZ;
    if ($g_i < ($grid_i-2)){
      $bXnext = ($posXCopy[$i+1]+$posXCopy[$i+2])/2;
      $bYnext = ($posYCopy[$i+1]+$posYCopy[$i+2])/2+0.25;
      $bZnext = ($posZCopy[$i+1]+$posZCopy[$i+2])/2-0.5;
      polyCreateFacet
        -p $bX $bY $bZ
        -p $posXCopy[$i+1] $posYCopy[$i+1] $posZCopy[$i+1]
        -p $bXnext $bYnext $bZnext;
    }
    select -clear;
  }
  if (($g_j == ($grid_j-1)) && ($g_i < ($grid_i-1))){
    $bX = ($posXCopy[$i]+$posXCopy[$i+1])/2;
    $bY = ($posYCopy[$i]+$posYCopy[$i+1])/2+0.25;
    $bZ = ($posZCopy[$i]+$posZCopy[$i+1])/2+0.5;
    polyCreateFacet
      -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
      -p $posXCopy[$i+1] $posYCopy[$i+1] $posZCopy[$i+1]
      -p $bX $bY $bZ;
    if ($g_i < ($grid_i-2)){
      $bXnext = ($posXCopy[$i+1]+$posXCopy[$i+2])/2;
      $bYnext = ($posYCopy[$i+1]+$posYCopy[$i+2])/2+0.25;
      $bZnext = ($posZCopy[$i+1]+$posZCopy[$i+2])/2+0.5;
      polyCreateFacet
        -p $bX $bY $bZ
        -p $posXCopy[$i+1] $posYCopy[$i+1] $posZCopy[$i+1]
        -p $bXnext $bYnext $bZnext;
    }
    select -clear;
```

```
}
if (($g_i == 0) && ($g_j < ($grid_j-1))){
  $bX = ($posXCopy[$i]+$posXCopy[$i+$grid_i])/2-0.5;
  $bY = ($posYCopy[$i]+$posYCopy[$i+$grid_i])/2+0.25;
  $bZ = ($posZCopy[$i]+$posZCopy[$i+$grid_i])/2;
  polyCreateFacet
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
    -p $posXCopy[$i+$grid_i] $posYCopy[$i+$grid_i] $posZCopy[$i+$grid_i]
    -p $bX $bY $bZ;
  if ($g_j < ($grid_j-2)){
    $bXnext = ($posXCopy[$i+$grid_i]+$posXCopy[$i+2*$grid_i])/2-0.5;
    $bYnext = ($posYCopy[$i+$grid_i]+$posYCopy[$i+2*$grid_i])/2+0.25;
    $bZnext = ($posZCopy[$i+$grid_i]+$posZCopy[$i+2*$grid_i])/2;
    polyCreateFacet
      -p $bX $bY $bZ
      -p $posXCopy[$i+$grid_i] $posYCopy[$i+$grid_i] $posZCopy[$i+$grid_i]
      -p $bXnext $bYnext $bZnext;
  }
  select -clear;
}
if (($g_i == ($grid_i-1)) && ($g_j < ($grid_j-1))){
  $bX = ($posXCopy[$i]+$posXCopy[$i+$grid_i])/2+0.5;
  $bY = ($posYCopy[$i]+$posYCopy[$i+$grid_i])/2+0.25;
  $bZ = ($posZCopy[$i]+$posZCopy[$i+$grid_i])/2;
  polyCreateFacet
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
    -p $posXCopy[$i+$grid_i] $posYCopy[$i+$grid_i] $posZCopy[$i+$grid_i]
    -p $bX $bY $bZ;
  if ($g_j < ($grid_j-2)){
    $bXnext = ($posXCopy[$i+$grid_i]+$posXCopy[$i+2*$grid_i])/2+0.5;
    $bYnext = ($posYCopy[$i+$grid_i]+$posYCopy[$i+2*$grid_i])/2+0.25;
    $bZnext = ($posZCopy[$i+$grid_i]+$posZCopy[$i+2*$grid_i])/2;
    polyCreateFacet
      -p $bX $bY $bZ
      -p $posXCopy[$i+$grid_i] $posYCopy[$i+$grid_i] $posZCopy[$i+$grid_i]
```

```
      -p $bXnext $bYnext $bZnext;
   }
   select -clear;
  }
  if (($g_i == 0) && ($g_j == 0)){
   $bX = ($posXCopy[$i]+$posXCopy[$i+$grid_i])/2-0.5;
   $bY = ($posYCopy[$i]+$posYCopy[$i+$grid_i])/2+0.25;
   $bZ = ($posZCopy[$i]+$posZCopy[$i+$grid_i])/2;
   $bXnext = ($posXCopy[$i]+$posXCopy[$i+1])/2;
   $bYnext = ($posYCopy[$i]+$posYCopy[$i+1])/2+0.25;
   $bZnext = ($posZCopy[$i]+$posZCopy[$i+1])/2-0.5;
   polyCreateFacet
    -p $bX $bY $bZ
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
    -p ($posXCopy[$i]-0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]-0.5);
   polyCreateFacet
    -p $bXnext $bYnext $bZnext
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
    -p ($posXCopy[$i]-0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]-0.5);
   select -clear;
  }
  if (($g_i == ($grid_i-1)) && ($g_j == 0)){
   $bX = ($posXCopy[$i]+$posXCopy[$i-1])/2;
   $bY = ($posYCopy[$i]+$posYCopy[$i-1])/2+0.25;
   $bZ = ($posZCopy[$i]+$posZCopy[$i-1])/2-0.5;
   $bXnext = ($posXCopy[$i]+$posXCopy[$i+$grid_i])/2+0.5;
   $bYnext = ($posYCopy[$i]+$posYCopy[$i+$grid_i])/2+0.25;
   $bZnext = ($posZCopy[$i]+$posZCopy[$i+$grid_i])/2;
   polyCreateFacet
    -p $bX $bY $bZ
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
    -p ($posXCopy[$i]+0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]-0.5);
   polyCreateFacet
    -p $bXnext $bYnext $bZnext
    -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
```

```
    -p ($posXCopy[$i]+0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]-0.5);
   select -clear;
 }
 if (($g_i == 0) && ($g_j == ($grid_j-1))){
   $bX = ($posXCopy[$i]+$posXCopy[$i-$grid_i])/2-0.5;
   $bY = ($posYCopy[$i]+$posYCopy[$i-$grid_i])/2+0.25;
   $bZ = ($posZCopy[$i]+$posZCopy[$i-$grid_i])/2;
   $bXnext = ($posXCopy[$i]+$posXCopy[$i+1])/2;
   $bYnext = ($posYCopy[$i]+$posYCopy[$i+1])/2+0.25;
   $bZnext = ($posZCopy[$i]+$posZCopy[$i+1])/2+0.5;
   polyCreateFacet
     -p $bX $bY $bZ
     -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
     -p ($posXCopy[$i]-0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]+0.5);
   polyCreateFacet
     -p $bXnext $bYnext $bZnext
     -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
     -p ($posXCopy[$i]-0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]+0.5);
   select -clear;
 }
 if (($g_i == ($grid_i-1)) && ($g_j == ($grid_j-1))){
   $bX = ($posXCopy[$i]+$posXCopy[$i-$grid_i])/2+0.5;
   $bY = ($posYCopy[$i]+$posYCopy[$i-$grid_i])/2+0.25;
   $bZ = ($posZCopy[$i]+$posZCopy[$i-$grid_i])/2;
   $bXnext = ($posXCopy[$i]+$posXCopy[$i-1])/2;
   $bYnext = ($posYCopy[$i]+$posYCopy[$i-1])/2+0.25;
   $bZnext = ($posZCopy[$i]+$posZCopy[$i-1])/2+0.5;
   polyCreateFacet
     -p $bX $bY $bZ
     -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
     -p ($posXCopy[$i]+0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]+0.5);
   polyCreateFacet
     -p $bXnext $bYnext $bZnext
     -p $posXCopy[$i] $posYCopy[$i] $posZCopy[$i]
     -p ($posXCopy[$i]+0.5) ($posYCopy[$i]+0.25) ($posZCopy[$i]+0.5);
```

```
      select -clear;
    }
  }
}
float $moveX = $sizeX-$relX+2;
float $moveZ = $sizeZ-$relZ+2;
polyCube -w $moveX -h $thickness -d $moveZ;
move -r (($moveX-2)/2) (-1*$thickness/2) (($moveZ-2)/2);
select -clear;
// -----------
for ($i=0; $i<(2*$numberAttractor); $i){
  int $index = $attractor[$i]+$attractor[$i+1]*$grid_i;
  sphere
    -pivot $posX[$index] $posY[$index] $posZ[$index]
    -radius 0.4;
  move -r ($sizeX+10) 0 0;
  select -clear;
  $i=$i+2;
}
for ($i=0; $i<size($distance); $i++){
  int $iConnect = $connectTo[$i];
  if ($iConnect != $i){
    sphere
      -pivot $posX[$i] $posY[$i] $posZ[$i]
      -radius 0.2;
    move -r ($sizeX+10) 0 0;
    sphere
      -pivot $posX[$iConnect] $posY[$iConnect] $posZ[$iConnect]
      -radius 0.2;
    move -r ($sizeX+10) 0 0;
    polyCreateFacet
      -p $posX[$i] $posY[$i] $posZ[$i]
      -p $posX[$iConnect] $posY[$iConnect] $posZ[$iConnect]
      -p ($posX[$iConnect]-0.05) ($posY[$iConnect]-0.05) ($posZ[$iConnect]-0.05)
      -p ($posX[$i]-0.05) ($posY[$i]-0.05) ($posZ[$i]-0.05);
```

```
    move -r ($sizeX+10) 0 0;
    select -clear;
  }
}
}
```

caad DARCH
Prof. Dr. Ludger Hovestadt
Computer Aided Architectural Design

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich