EMERGENT PROGRAMATIC FORM-ATION

Formulating a Responsive Process of Designing Architecture

by

Yehia Madkour

A THESIS SUBMITTED IN PARTIAL FULLFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ADVANCED STUDIES IN ARCHITECTURE

in

The Faculty of Graduate Studies

(Architecture)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

April 2009

Abstract:

The increasing use of digital techniques in architectural design clearly exhibits the temptation to develop geometries of formal complexity. However, the conditions that influence form-making decisions are often too abstracted to address specific pragmatic references. Parametric modeling offers an approach to design capable of controlling multiple processes and transformations. On the other hand, it is still a challenge for the architect to deploy the particular conditions of context, user functions, and program based on a complex set of interrelated natural and social references in a parametric design system.

This thesis illustrates how parametric modeling can be productively integrated into the current design and planning process in an urban setting. A series of explorations demonstrate parametric modeling's potential to engage and coordinate a complex set of parameters that are used to integrate particular performance criteria into the digital design process at a variety of scales. This approach formulates a shift in the digital design process away from the ambiguity of formal complexity to a focus on programmatic spatial configurations. In each of the explorations, dynamic dependencies are created between the model and its influencing programmatic factors to develop a responsive interaction between the designer, the design model, and the references it reflects.

Table of Contents:

Abstract ii

Table of Contents iii

List of Figures iv

Acknowledgements vii

Definitions viii

Chapter 1: Introduction 1

Chapter 2: Technique 18

Chapter 3: Space Configuration 32

Chapter 4: Tower Formation 42

Chapter 5: Shade 72

Chapter 6: City Configurations 105

Chapter 7: Conclusion 124

Bibliography 148

Figures Credits 151

List of Figures:

1.1 Methodology Diagram 6
1.2 Game Tree for Tic-Tac-Toe 11
2.1 Son O House 22
2.2 Son O House Process 23
2.3 Log Cabin Process 1 27
2.4 Log Cabin Process 2 28
2.5 Pneumatic Strawberry Bar Process 30
2.6 Comparison of Existing Digital Techniques 32
3.1 Existing Social Condition in downtown Vancouver 42
3.2 Typical Living Unit 43
3.3 Sub-spaces of Living Unit 45
3.4 Components of Living Unit 45
3.5 Model Unit 46
3.6 Model Variables 47
3.7 Design Initiation 49
3.8 Possible Households 49
3.9 Social System Parameters 50
3.10 Environmental System Parameters 52
3.11 Space Configurations System Making 53
3.12 Space Configurations System Making 54
3.13 Space Configurations System Making 55
3.14 Space Configurations System Making 56
3.15 Space Configurations System Making 57
3.16 Space Configurations System Making 58
3.17 Space Configurations System Making 59
3.18 Space Configurations System Making 60
3.19 Space Configurations System Making 61
3.20 Space Configurations System Making 62
3.21 Space Configurations System Making 63
3.22 Space Configurations System Making 64

3.23 Instance of Model Configuration 65
3.24 Living Scenario 1 67
3.25 Living Scenario 2 68
3.26 Living Scenario 3 69
4.1 Site 76
4.2 Existing Architecture on Site 77
4.3 View Corridor 9.2.1 79
4.4 Context at Form Scale 1 80
4.5 Context at From Scale 2 80
4.6 Orientation 81
4.7 Tower Formation Process 82
4.8 Tower Formation System Making 83
4.9 Tower Formation System Making 84
4.10 Tower Formation System Making 85
4.11 Tower Formation System Making 86
4.12 Tower Formation System Making 87
4.13 Tower Formation System Making 87
4.14 Tower Formation System Making 88
4.15 Tower Formation System Making 89
4.16 Tower Formation System Making 89
4.17 Tower Formation System Making 90
4.18 Tower Formation System Making 91
4.19 Tower Formation System Making 92
4.20 Tower Formation System Making 92
4.21 Tower Formation System Making 93
4.22 Tower Formation System Making 95
4.23 Tower Formation System Making 95
4.24 Tower Formation System Making 97
4.25 Tower Formation System Making 98
4.26 Tower Formation System Making 99
4.27 Tower Formation System Making 99
4.28 Tower Formation System Making 100

4.29 Tower Formation System Making 100
4.30 Tower Formation System Making 101
4.31 Tower Formation System Making 102
4.32 Tower Formation System Making 102
4.33 Physical Model of Tower Fromation 103
5.1 Unite d‘ Habitation, Marseilles, France...107
5.2 Base Shape 108
5.3 Shadying Sytem 108
5.4 Shade System Making 109
5.5 Shade System Making 109
5.6 Shade System Making 110
5.7 Shade System Making 111
5.8 Shade System Making 111
6.1 City of Vancouver guidelines and regulations....................................127
6.2 Fixed Density 131
6.3 Shading study 132
6.4 Shading Impact...132
6.5 View Blocking Test 133
6.6 City Configurations Parameters 134
6.7 City Configurations system Making..135
6.8 City Configurations system Making..136
6.9 City Configurations system Making..137
6.10 City Configurations system Making..138
6.11 City Configurations system Making..139
6.12 City Configurations system Making..139
6.13 City Configurations system Making..140
6.14 City Configurations system Making..140
6.15 City Configurations system Making..142

Acknowledgements:

I am grateful to my thesis committee, Oliver Neuman, Matthew Soules and Halil Erhan for their ensightful comments, feedback, and support throughout my thesis work. I would like to thank Sherry Mckay for her guidance since my start at the MASA program. I would like to thank Dina Krunic for introducing me to computational design in her studio in 2005. I would like to thank Greg Johnson for his conceptual contribution in the context of the designed shading system. I would like to thank all the faculty and staff at the UBC School of Architecture, who collectively contributed to a great educational experience. I would like to thank Tarek Madkour, my brother, for introducing me to computer programming at the beginning of my research. I am grateful to my boss, Wilson Chang who has shown repeated consideration in allowing me the time to continue my research.

Without my friends and colleagues at the School of Architecture, it would have not been possible to complete the thesis. Thank you specially to Sanaz Mani, Guadalupe Font, Alan Bushby, Farshad Jamali, Mohamed Magdy, and Shankari Reebye.

Special thanks to my wife Riham, who has provided me with much needed understanding and support. And last but not least, thank you to Narmine Aly and Ashraf Madkour, my parents who I would like to dedicate my thesis to, for their immense emotional and financial support.

Definitions:

Algorithm : A set of well-defined rules (procedure) for the solution of a problem in a finite number of steps.

Computation: Any type of information processing that can be represented by a sequence of operations. This includes phenomena ranging from human (logical) thinking to mathematical calculations.

Responsive: The ability to dynamically respond (react) to different criteria in the design problem.

System: A set of interacting or interdependent elements designed to work as a coherent entity.

1 Introduction

Introduction

The increasing use of digital techniques in design clearly exhibits the instant and inevitable temptation to develop complex formal solutions. However, the drivers that influence form-making decisions are often too abstracted to address specific references. Although parametric modeling holds rich potential for controlling multiple processes and transformations, it is a bigger challenge and responsibility to employ the digital design techniques in designs that are responsive to the particular conditions of context, user functions, and program based on a complex set of interrelated natural and social references.

Emergent Programmatic Form-ation illustrates how parametric modeling can be productively integrated into the current design and planning process to coordinate complex relationships and conditions specific to urban housing.

Chapter 1 reviews key theories that analyse how natural and artificial systems operate. In particular, it provides the conceptual background and methodology that defines the computational and parametric approach to design proposed and applied throughout the thesis.

Chapter 2 presents three existing computational design techniques and precedent projects. The chapter formulates the project's parametric design approach, and proposes a shift in the digital design process away from formal ambiguity to a focus on spatial configurations that reflect a complex array of programmatic performance criteria.

Chapters 3 to 6 present a series of parametric design explorations. The explorations define new grounds where digital techniques manipulate not only form, but also function, program and space. In each of the explorations, dynamic relationships are created between the model and its influencing programmatic factors, where the digital model stores explicit performance-sensitive design decisions and constraints and responds to them simultaneously. This allows for a complex and cumulative expression, both programmatic and spatial:

Chapter 3: '*Space Configurations'* uses social and environmental criteria to generate a wide range of spatial configurations and modes of occupation within a typical city apartment;

Chapter 4: '*Tower Formation'* explores how the physical and environmental context affects the design formation of a residential tower inside the city core;

Chapter 5: '*Shade'* is a responsive sun shading system that calculates its cardinal orientation and accordingly controls the lengths of horizontal and vertical overhangs;

Chapter 6: '*City Configurations'*, the last exploration, engages the city at a larger scale and explores a responsive way of planning and regulating the relationships between buildings in the city in an attempt to establish a dynamic base that incorporates a complex set of city planning guidelines.

Chapter 7 presents conclusions driven from the explorations of the previous chapters.

Objective:

The goal of this thesis is to highlight the interaction between designers and the computer tool through the integration of computation in a pragmatic design process. Computational design techniques are applied on a series of explorations of varying program and scale to establish architecture that is responsive to programmatic and context sensitive criteria through the design process.

Hypothesis:

The thesis explores the hypothesis that computation and digital techniques can help inform the design process. Through the formation of architecture, the thesis proposes computational and parametric design systems that respond to varying pragmatic criteria that address a complex set of interrelated natural, social, and economic references. Parametric modeling holds rich potential for controlling multiple processes and transformations. In this thesis, the main goal is to explore the potential of parametric modeling to dynamically resolve and control the particular conditions of context, user functions, and program that define architecture projects.

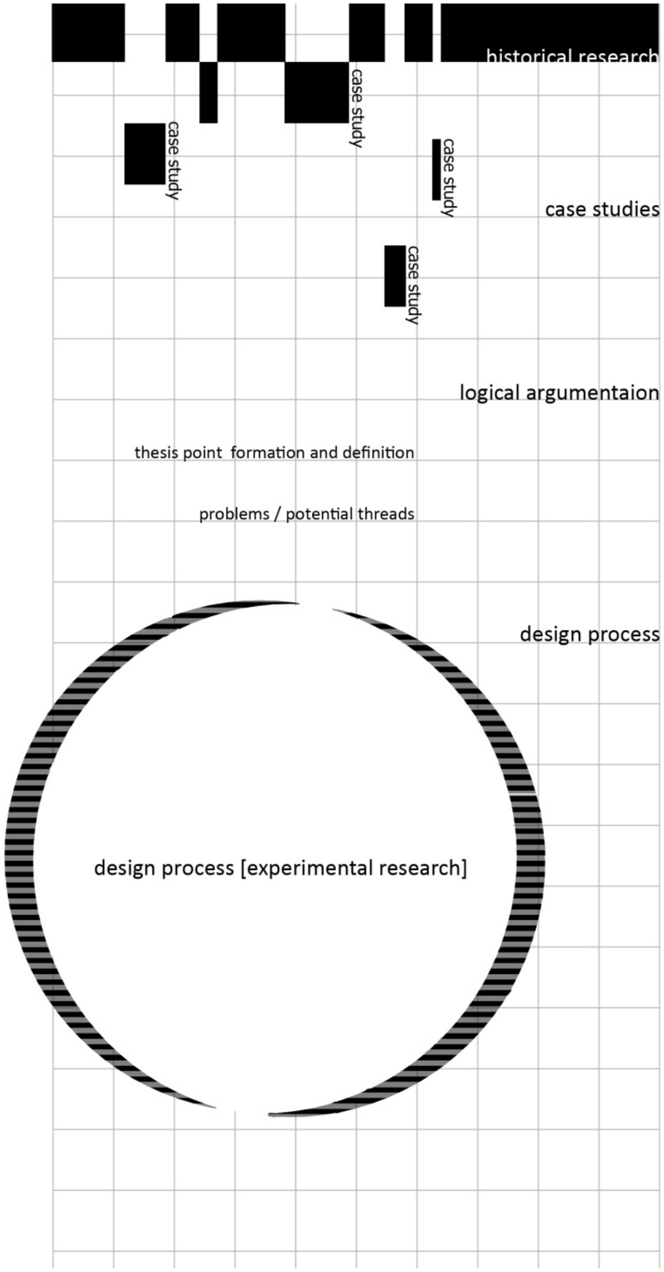
Contribution:

The key contribution of this thesis is the original use of context, project program, and user-sensitive design drivers within the digital design field. In a series of design explorations performed in the third through the sixth chapters of this thesis, form making is influenced by and responsive to programmatic and functional issues arising from each exploration's design problem. A study of relevant architectural design precedents illustrated in chapter two reveals a prominent disregard of program-sensitive approaches to design problems. To date, the drivers that influence form-making decisions are often too abstracted to address programmatic design issues. A bigger challenge is in designing spaces considering issues related to user functions and project programs, compared to producing complex open space mega structures and surface geometries. User-oriented programmatic functions hold many limitations for the production of form; which, if accompanied by the still limited degree of control associated with algorithmic techniques, explains the disregard of program-sensitive design approaches in the field.

Urban Housing:

Housing design in metropolitan areas is used as a platform for investigating computational techniques in the architectural design process. The thesis explores parametric modeling techniques to incorporate and coordinate complex relationships and conditions specific to urban housing. User-oriented programmatic functions based on required performance criteria, typical standard sizes and proportions of domestic spaces, efficiency, user preferences, as well as the natural and artificial context and their influence on buildings, all hold limitations for the production of form. In the proposed parametric design systems, the multiple design criteria and limitations are used as the design drivers and constraints on which the systems operate and to which they respond.

The widely used method of prototyping uses and accordingly prototyping architectural form is strongly criticised throughout the design work in this thesis. As users are different in their needs and the way they live and interact, so have the spaces that accommodate them to be different. The



1.1 Methodology Diagram

technique to design myriad varieties of housing options is obviously different from that needed to produce prototypes. The project explores the capacity of parametric modeling to coordinate a wide range of pragmatic parameters. It is argued in the thesis that being able to manipulate multiple processes via a computational design is the method to avoid architecture prototyping and better address the interrelated needs.

Methodology

The thesis strategically establishes its position between design and research by integrating episodes of research throughout the design process. The project starts with historical research to comprehend the background theories informing computational design approaches. Case studies of relevant design projects are evaluated through logical argumentation, which helped defining the thesis statement as well as the proposed design approach. A series of design explorations are performed to explore the capacity of parametric design to coordinate a wide range of performance-related design parameters in an architectural design process. Throughout each of the explorations, action research and reflective assessment are used to plan and operate the proposed computational design systems. The work in this design-research thesis progresses through consecutive episodes of research and explorations to produce design systems that are dynamically responsive to their functional context, in a way it informs designers through the process.

History: My historical research starts with an overview of key theories significant to digital technologies and their applications in designing architecture. Complexity, Emergence, and Natural Systems are the main key points in my historical theory overview. I used these concepts to formulate my proposed design approach.

A detailed study of digital design techniques used in architectural projects reveals huge transformation in the applications of computers in the design of architecture. An understanding of the existing digital design techniques and their potential and limitations influenced the decision for choosing parametric modeling as a proposed design technique.

Case Studies: Emerging out of a detailed historical study, the case studies phase is a look at prominent architectural precedent projects that utilize digital systems in the process of form generation. Pre-programming the design process by studying existing related case studies and evaluating them is a necessary tactic to gather information about the process and spot potential problems. Besides opening an eye on the existing techniques, this study revealed the architects' general temptation to focus on formal solutions when computation is involved in design. A disregard to function and program sensitive approaches became obvious in this study.

Logical Argumentation: This phase builds upon the understanding and analysis of preceding phases. The thesis develops a critical stand towards the selected case studies, which leads to identifying a potential research area in the use of context and program-sensitive design drivers within the digital design field. In other words, the logical argumentation phase is critical to the definition of my thesis research.

Design process: The process starts by addressing emerging problematic issues within residential architecture in the city. These issues direct the flow of experimental research, which is undertaken through a series of design explorations that aim to find the possibilities computers could offer to the design process. Through the entire design exploration series, action research and reflective assessment are important to extract knowledge and inform the experiments; by means of diagnosing the design problems, action planning, and the strategic translation of performance-related design constraints to parameters that manipulate and control the computational model. The computational design systems proposed in each of the explorations incorporate the influential design constraints into the design model, creating a dynamic link of reflective assessment, which constantly feeds information between the design and its influences within the local exploration, allowing the designer to instantly evaluate the generated design outputs. This approach ultimately leads each exploration to inform the succeeding ones.

Background theory

The thesis builds on a number of ideas and theories that analyse how natural and artificial systems around us operate. In particular, the thesis draws its computational approach to design on the conceptual understanding of these systems.

Connected to a world reference that is continuously changing and adapting, the architecture scene has much yet to be understood and appreciated. Much of the current design approaches remains committed to outdated references and models. We encounter how technology has dramatically transformed social interactions and caused a global political change. To overcome the outdated references and models, it is necessary to develop a more sophisticated account of the complex and dynamic evolution of the natural, cultural, political and economical processes. The way we live and experience these adaptive systems should prompt us to change the way we design our living.

Complexity and Emergence are the key theories that provide a conceptual understanding of the surrounding adaptive systems. They also serve as the inspiration for the computational design processes shown in the next chapters.

Complexity:

The world around us exhibits complexity at all levels, from ant colonies, to the immune system, to the interdependencies of financial markets and political parties, to the ecologies formed by living things. Each system is highly organised and defined by more simple laws that govern the interaction between its elements. Complex Systems is an approach to science that studies the relationship between the individual elements or parts, how their interaction leads to a collective behaviour of a system, and how this system responds to its environment.

Today’s world: A look at today’s urban environments reveals places marked by natural, cultural, political, and economical infusions. These infusions show a complex interconnectivity where not only people meet, but also ideas, images, and products flow freely throughout the world. Today's market is controlled by network economy, where the notion of modernist's fixed price in a stable system is no longer applied. Network economy is composed of many different parts that interact in multiple ways to generate effects and events, and though generated by local interactions; the effects tend to be global. These interactions contribute to a complex system that, although controlled and programmed by governing laws, displays spontaneous self-organization that is not pre-planned.

What describes today's economy can be applied to many of the phenomena that mark today's city. World politics displays an example where a dynamic adaptive system is created and controlled by many political micro parties that trigger major effects on the global scale. The system is never stable as long as its elements continue to change, but it manages to display self-organization on the macro scale. The internet is a third example of complex adaptive systems. It is a publicly accessible series of interconnected computer networks, which consist of millions of smaller domestic, academic, business, and government networks. The internet displays a fractal self-organized structure; its organic growth is controlled by the behaviour of the microstructures. Iterations at the smaller scale generate a macrostructure, and the operation of the macrostructure sustains the microstructure.

Definition: A Complex System could be defined as a system composed of interconnected elements that as a whole exhibit properties not obvious from the properties of the individual parts. Special cases of complex systems are Complex Adaptive Systems (CAS); they are adaptive in that they have the capacity to change and learn from experience.

Static Models: Studying complex systems heavily relies on 'reduction'. Complex Systems are described in terms of interaction of simpler components. Building scale models is a good method to reduce a system to its essential information and make it easier to comprehend and manipulate. This method works for conventional design approaches as well as computational ones. A model serves to simplify the building and highlight the important elements relevant to the design stage in order to inform design decision making. Shearing away detail is an essence of model making; a model must be simpler than the object modeled.

However, knowing the behaviour of simple isolated parts leaves us a long way from understanding the whole. The notion of reduction by studying isolated parts is not solely effective in studying complex systems. We have to understand the interactions as well as the parts.

Emergence:

> *"We are everywhere confronted with emergence in complex adaptive systems -ant colonies, networks of neurons, the immune system, the internet, and the global economy, to name a few - where the behaviour of the whole is much more complex than the behaviour of the part."* (Holland 2).

Emergence acts as a good description of such systems, where the activities of parts do not simply sum to give the activity of the whole. The above quote is from the opening chapter of 'Emergence from Chaos to Order'. Holland's work is very effective in defining technical concepts as a foundation for studying emergence.

Game Theory: A basic definition of emergence is *much coming from little.* Holland interestingly uses board games as a simple example of the emergence of complexity from simple rules or laws. A number of rules are set to define a board game, and every time a round passes, the arrangements of pieces on the board continuously grow into a more complex state, and state results come from player's actions that follow the same set of initial rules. We cannot simply understand the state of a

Figure 1.2 has been removed due to copyright restrictions. The information removed include part of a Tic Tac Toe game tree.

Holland, John. Emergence: From Chaos to Order. Basic Books, 1998, 37

1.2 **Game Tree for Tic Tac Toe**

game in progress by understanding the state of individual players on theboard. The players, guided by the rules of the game, interact to support one another and to control various parts of the game. The power of relationships continuously changes the state of the game without being pre-planned. The state of the game quickly rises to complexity because it is animated and dynamic; it changes over time although the guiding laws are invariant.

Dynamic Models: In contrast to models with static form, such as scale models, dynamic models change in configuration. The key in constructing dynamic models is finding unchanging laws that generate the configuration change and define the interaction between smaller components of the model. These laws correspond roughly to the rules of a game that set the legal way of moving or placing the pieces on the board. Using this logic, we can construct models that exhibit dynamic emergent phenomena, much like the way interacting strategies in a game produce patterns of interaction not easily understood from the review of the rules of the game.

Natural systems: Growth in natural systems is a great example of complexity and emergence. It is illustrated when you think of plant seeds; these small capsules enclose specifications that produce complex and distinctive plant structures. A seed encapsulates the genes which specify step-by-step unfolding of biochemical interactions 'genotype'; it not only states the shape of the plant but also the way the genes interact with its surrounding in order to grow and survive 'phenotype'.

The 'phenotype' and 'genotype' terms were used by Una-May O'Reilly, Martin Hemberg, and Achim Henges to describe the complex process of natural growth in their attempt to explore the potential of generative algorithms as operative design tools. The goal of their exploration was " .. to instrumentalise the natural processes of evolution and growth, to model essential features of emergence and then to combine these within a computational framework" (O'Reilly 49). Their aim is to apply this instrument as a generative design tool that can produce complex architectural forms.

The understanding of emergence provides both an explanation of how natural systems evolve and maintain themselves, and a process for the creation of artificial systems that are designed to produce forms of complex behaviour. These systems are greatly demonstrated in Guilles Deleuze's arguments that matter has powers of morphogenetic form, a never-ending series of exchange between a system and its environment. [1]

This argument clarifies the form-finding experiments of Frei Otto to resolve lightweight structures. In 'Frei Otto in Conversation with the Emergence and Design Group', Otto discusses his development of modelling through form-finding techniques in the context of his interest in natural systems. Soap bubbles and spider webs give inspiring examples in the behaviour of form-finding lightweight structures. Otto's work demonstrates how forms were generated by analysing natural growth systems.

Emergence in architecture: If we understand design as a process of emergence where a generative production of forms exists, strategic techniques in can be explored to model the process and reconstruct existing design methodologies. Emergence, in a more precise definition, is the creation of artificial systems that are designed to produce forms of complex behaviour, where a system is set to generate complexity out of simple initial states. Laws for generating emergence can be set by designers to address specific design problems, and executed as procedures through a designed system. A computational framework helps in modeling emergence, because the calculation power offers a great way to simultaneously address several design problems and dynamically represent the accumulated effects for the designer to explore. The defined laws or rules act as constraints that guide the process of design.

Architecture, though typically considered a static art, can play a role in the awareness of surrounding dynamics of complex adaptive systems that influence architecture and its design. Architects need to rethink how architecture is designed, and develop a methodology that takes into account the dramatic change in how the world operates. I see the application of emergence concepts in architecture as a design process in which the architect necessarily thinks of a building as a complex set of relationships, and the design process is a design of a system that is capable of evolving to higher degrees of complexity.

It is not necessary that a building itself is viewed like the living organism, but it is the design procedure that has to evolve this way because at the end a single design is captured and executed as a single element. Helen Castle, in 'Emergence in Architecture', calls for constructing architecture that has evolved through a process of morphogenesis, rather than regarding buildings as unchanging, isolated tectonic objects. A similar stand was put forward by the ‘Emergence and Design Group’ in their argument:

> *'Emergence requires the recognition of buildings not as single bodies, but as complex energy and material systems that have a life span, and exist as part of the environment of other buildings, and as an iteration of a long series*

> *that proceeds by evolutionary development towards and intelligent ecosystem'*. (O'Reilly 51)

We can design systems and open them to manipulation by modeling the essential features of emergence. These systems are needed to introduce a responsive architecture at model scale, which helps in understanding how the interaction of varying social, cultural, and economical parameters can affect spaces. Being able to respond to the dynamic influences require design techniques that can tackle the involved complexity. The logic and essential features of emergence leads the path to solving complex design problems.

However, morphogenetic processes and genetic breeding systems are oftentimes perceived as a must-have association. In my position, this approach to design relies on the understanding of the basic rules that exist in the growth of natural systems, and the key is applying the main conceptual logic to conduct growth out of designed rules. It is not necessarily associated with the evolutionary breeding of design populations and choosing the 'fittest' to survive.

It quickly becomes clear that the concept of emergence relies on blurring the boundaries between once quite separate sciences; overlapping domains of developmental biology, physical chemistry, mathematics, and computer science are now urged to find their application to architecture. The convergent lines of thought between biology and mathematics were initiated early in the 20th century, particularly in the work of D'Arcy Thompson where he regarded the material forms of living organisms as a diagram of the forces that acted on them to cause the morphing action between species[2]. In 'Digital Morphogenesis', Neil Leach argues for the need of using theoretical and scientific work from other domains to influence the idea morphogenetic architectural design[3]. He believes that Guilles Delueze's theories on material behaviour and D'Arcy Thompson's theory of transformation will possibly shift architecture practice, in that it observes complexity as it emerges from a simple set of rules.

Design Approach

The thesis proposes a design approach that uses rule-based procedures to generate complexity out of simple initial states.

A rule-based design system is the key to building dynamic models that respond to changing influences. The systematic unfolding of programmed instructions allows for continuous interruption and recalculation of the building model. The responsive and adaptive character of such a system makes it relatively easy to examine the effects of manipulations. This design approach potentially allows responsive relationships to exist between initial design influences and the designed spaces.

Computers are essential to the deployment of rule-based design techniques because of their vigorous calculation power that is capable of executing the complex procedures required to address design complexity.

Subjectivity:

What makes this design process problematic for architects is that they have maintained a notion of artistic sensibility and creativity in their practice. In contrast, a computer algorithm is considered to be a non-human creation and therefore distant and remote. Because of this debate about design subjectivity and innovation, the key point is in how the designer uses the algorithmic logic. It is important that the designer have the desired control over the articulation of setting the rules. The designer sets the parameters of the design game, and has the opportunity to interpret and control the process. On the other hand, this approach to design draws an objective methodology of form-finding through its mere execution of rules.

[1] DeLanda, Manuel. "Deleuze and the Use of Genetic Algorithms in Architecture." Architectural Design January 2002: 9-12.

[2] Thompson, D'Arcy Wentworth. On Growth and Form. Cambridge University Press, 1942.

[3]

Leach, Neil. "Digital Morphogenesis: A New Paradigmatic Shift in Architecture." Archithese 2006: 44-49.

2 Design Technique

Computers in Architecture:

The application of the proposed rule-based design approach needs vigorous calculation power that is capable of executing complex procedures. Computers are a necessary tool for the realisation of such a design approach. To architects, computers should be more than a sophisticated drafting tool; these are device with potential to be part of the design process itself. However, the current and conventional use of software in the architecture profession limits the possibilities of computers. A collection of CAD and graphics software is largely used to present pre-designed building models, but is not usually integrated into the design process itself. Architects tend to design by conceiving spatial forms after the consideration of the project's program, context and other design-influencing factors. These considerations are intermingled with designers' conception and vision, and are ultimately translated into building models. Plans, elevations, and sections are examples of building models, as are renderings, virtual and physical scale three-dimensional models. Computer software packages are used in this case to produce representations of conceived spaces. The link between the designed spaces and initial influencing factors is entirely conceived in the black box of the designer's mind. Parametric modeling can be used to explore the ambiguous design process outside of the designer's mind by means of incorporating the design influences and constraints into the model. Scripting and parametric modeling makes possible the application of mathematical techniques for modeling the emergence of forms and the complex behaviour of dynamic responsiveness.

'We can use the mathematical models outlined above for generating designs, evolving forms and structures in morphogenetic processes within computational environments' (Weinstock 17)

In principal, we could use pencil and paper to carry out the instructions of any such program. However, the more complicated the task is, the more the process becomes time-consuming beyond feasible human capability. Computers offer a speed that allows us to explore dynamic models, which display the state of a building in a certain level of abstraction, and allows for its manipulation.

Abstraction is a key element in building models. A system has to be reduced to its essential information to make it easier to comprehend and manipulate. In parametric systems, numbers are used to describe and control the model's information, much like the control panel in a car: it models a complex state of the car at a certain level of detail. Numbers can describe the information in the control panel displayed in gauges, lights, and even the sophisticated positioning system is reduced to an array of dots which is represented as an array of 0's and 1's. Similarly, in parametric architecture models, numbers are used to represent and coordinate dimensions, angles, folds, volumes, etc.

Digital Design Techniques:

In this chapter, I review existing computational design techniques used by architects through case studies of precedent projects. The work done in this review is helpful to formulating the project’s parametric design approach and developing a critical stand by proposing a shift in the digital design process away from formal ambiguity to focus on spatial configurations that reflect programmatic references.

Three existing categories of computational design techniques in architecture are identified and analysed. These techniques share a rule-based approach to design applied using different methods. They have one thing in common: which is analysing the purpose of design, abstracting the design approach to a set of rules, and accordingly setting a systematic procedure for a design to emerge.

1. Analog Computing Processes

Key Characteristics:

1. Abstraction
2. Rule setting logic
3. Manual execution of rules
4. Physical modeling experimentation

The analog computing process of design demonstrates cases where a design approach is abstracted to a set of rules to be followed in an analog environment. This process is clearly demonstrated in the form-finding

Figure 2.1 has been removed due to copyright restrictions. The information removed includes a photograph for the Son O House project' exterior.

Spuybroek, Lars. NOX. Thames & Hudson, 2004.

2.1 Son O House

experiments of Frei Otto to resolve lightweight structures. In a conversation with the Emergence and Design Group[4], Otto discusses his development of modelling through form-finding techniques in the context of his interest in natural systems. Soap bubbles and spider webs give inspiring examples in the behaviour of form-finding lightweight structures. Otto was exploring analogue computation in an early stage. His work demonstrates how forms are generated by analysing natural growth systems. The analysis of surrounding forces and how a system responds and finding the laws that governs these responses is a rule based approach to design that does not necessarily rely on computer software to breed generations, but it relies heavily on the rule setting logic and way of thinking to generate forms.

Case study 1:

Title:	Son-O-House
Date:	2004
Location:	Son En Breughel, The Netherlands
Program:	Sculptural pavilion for interactive sound
Architect:	Lars Spurbroek (NOX)

Overview:

The Son-O-House structure is both an architectural and sound installation that allows people to participate in the composition of the sound. The sound work, made by composer Edwin Van Der Heide, is continuously generating new sound patterns activated by sensors picking up actual movements of visitors.

Lars Spuybroek describes this pavilion as 'a house where sounds live'

(Spuybroek), and perceives it as a structure that refers to living and the body movements that accompany habit and habitation. Departing from this point, the design for the structure is derived from a fabric of large and small-scale body movements, which form an arabesque of complex intertwining lines that is both a reading of movements on various body scales and a material structure. The complex surfaces are structured as a series of intersecting flat ribs that would support the outer skin; the flat ribs are arranged to tile a doubly curved surface cladding.

Process:

Lars Spuybroek used analog rule-base design techniques to represent movement on various body scales and thus determine his structure.

1. A camera is first set to record human movements at different scales. Large movements are detected in corridors, while small ones are recorded around desks, sinks, etc. (Figure 2.2)

Figure 2.2 has been removed due to copyright restrictions. The information removed includes the design process photographs for the Son O House project.

Spuybroek, Lars. NOX. Thames & Hudson, 2004.

2. His initial mapping of the movement of bodies led to further modeling studies with paper. Cuts of different lengths are done on paper bands: 'an uncut area corresponds with the bodily movement, a first cut through the middle corresponds with limbs, and finer cuts correspond with movements of the hands and feet' (Spuybroek).

3. White paper bands are then twisted into various configurations. He stapled the pre-informed paper paths together at the point where they have the most connective potential, and as a result, curvature emerges.

4. Gradual definition of spatial volumes is generated by sets of red parallel paper strips bent perpendicularly over and through the white path strips. In each of these models, form emerged in a non-predictable fashion resulting from the reactions of the materials used to physical forces (bending, twisting, etc).

5. Computers were used to digitalise the paper models, produce a computer-aided analysis of the forces and stresses of the resulting complex geometries, and process the designed geometries for digital fabrication.

Comments:

This project illustrates the analog computing approach to design. It is clear in how the initial design approach is abstracted into a number of rules to be manually processed. However, the process clearly lacks responsiveness and is less open to instant changes in the design rules. The manual execution of design rules is potentially time consuming when procedures that are more complex are involved.

2. Scripted Algorithms

Key Characteristics:

1. Abstraction

2. Rule setting logic

3. Translation into defined steps

4. Execution in a digital environment

The scripted algorithms technique in design is the most explicit in the way a design process is deduced into a procedure of rules. Generally, an algorithm is a set of well-defined rules for the solution of a problem. The rule-based approach to design highly corresponds to the logic behind setting scripted algorithms. This technique carries out the well-defined instructions in a digital environment, making use of the computer's fast processing. A very powerful attribute of this technique is the use conditionals and loops, which are a sequence of statements specified once but which may be carried out a specified number of times or until a specific condition is met. Looping allows for repetition (in design) with changing variables.

Case study 2:

Title:	Log Cabin
Date:	2005
Location:	Competition
Program:	Log Cabin
Architect:	Benjamin Aranda and Chris Lasch

In their book *'Tooling'*, Benjamin Aranda and Chris Lasch follow a rule-based design approach where they focus on the methods to deduce a design into a procedural thinking of rules. They describe rule-based design as a recipe for shape generation. They investigate recipes for algorithms used to simulate natural phenomenon, like packing, weaving, blending, and tiling and put the mathematical logic of their making into simple steps. Through a number of projects, they explore turning these rules into logics for construction. In this case study, they investigate a packing recipe for a competition project to design a log cabin.

Process:

1. The construction of a log cabin involves cutting trees and then stacking them. Aranda and Lasch investigated a reversed order to start this project. By cutting and then stacking, they found an emerging capacity of wood to change shape from circle to ellipse to bar. Structure and opacity are found to be controllable, all through stacking and packing.

2. A recipe for packing is created to control the organisation method for logs. The recipe is a few steps that describe the logic they used to design the packing scheme of logs for the cabin (Figure 2.3).

Figure 2.3 has been removed due to copyright restrictions. The information removed includes the design process photographs and the packing recipe steps for the Log Cabin project.

Spuybroek, Lars. NOX. Thames & Hudson, 2004.

2.3 Log Cabin Process 1

The recipe deduces the packing process into picking a random point, creating a shape of random size, and looping the process without intersecting the shapes. Although the recipe they provide for the packing algorithm lacks a termination statement that otherwise will continue forever, this method is very powerful in generating forms by deducing the process into a defined number of steps. Variation to form can be easily manipulated by changes in the algorithm or its variables.

3. A plan was designed for the cabin, and the architects applied the packing system on the cabin's facades, with changes in the parameters of size, position, and repetition. In this case, the south facade gradually opens up towards the gathering space to take advantage of the view. (Figure 2.4)

Figure 2.4 has been removed due to copyright restrictions. The information removed includes exterior renderings and plans for the Log
project.

Spuybroek, Lars. NOX. Thames & Hudson, 2004.

2.4 Log Cabin Process 2

Comments:

This project illustrates the importance of strategically setting the rules for a design algorithm. It makes good use of loops and the processing power of computers in the way the packing recipe is designed. The designers were able to achieve various organisation methods by the manipulation of the script's geometrical parameters. However, the scripted algorithm technique does not allow for a dynamic adaption to changes. The design progresses on its preset path once the rules are set. Changes to design are applied by deleting a previous model and re-running the script.

3. Mimicking Organic Growth

Key Characteristics:

1. Rule setting logic
2. Virtual environmental conditions
3. Population of designs
4. Fitness evaluation procedures
5. Execution in a digital computational environment

This technique in design uses generative algorithms to create complex artefacts. The production of these artefacts is generally inspired by natural systems growth and executed through procedures in virtual environmental conditions. The designers use specific software packages to develop surface geometries in a three dimensional space that has embedded modifying commands that simulate environmental conditions. The result of such technique is a production of population adaptations of geometries in many generations that follow the designer's rules. Instance of the design population are evaluated and chosen according to a fitness function that rules out invalid solutions.

Case study 3:

Title:	Pneumatic Strawberry Bar
Date:	2004
Location:	London, England
Program:	Strawberry Bar
Architect:	Achim Menges

This project is a small-scale design for a strawberry bar for the Architecture Association annual exhibition. The designer's basic idea is to use pure geometric evaluation criteria as a substitute for the structural analysis, to guarantee the structural integrity of the resulting folded form.

Figure 2.5 has been removed due to copyright restrictions. The information removed includes the design process and renderings for the Pneumatic Strawberry Bar project.

http://www.achimmenges.net/achimmenges_pl04_DesignExper.swf
Accessed 13 4 09

2.5 Pneumatic Strawberry Bar Process

Process:

1. The bar's design is a folded structure that starts with two trapezoids connected at their common seem.

2. A Maya plug-in, Genr8, is used to breed a population of the base trapezoidal shape elements according to geometrical fitness criteria that each has a target value and a fractional weight.

These criteria include:

a. Symmetry

b. Size

c. Soft boundaries: This determines if the populated surfaces are allowed to grow through the boundary wall.

d. Subdivisions: Measures the amount of articulation in the surface.

e. Smoothness: Measures the extent of local geometry variations in the Z plane. It controls the shape of the base components.

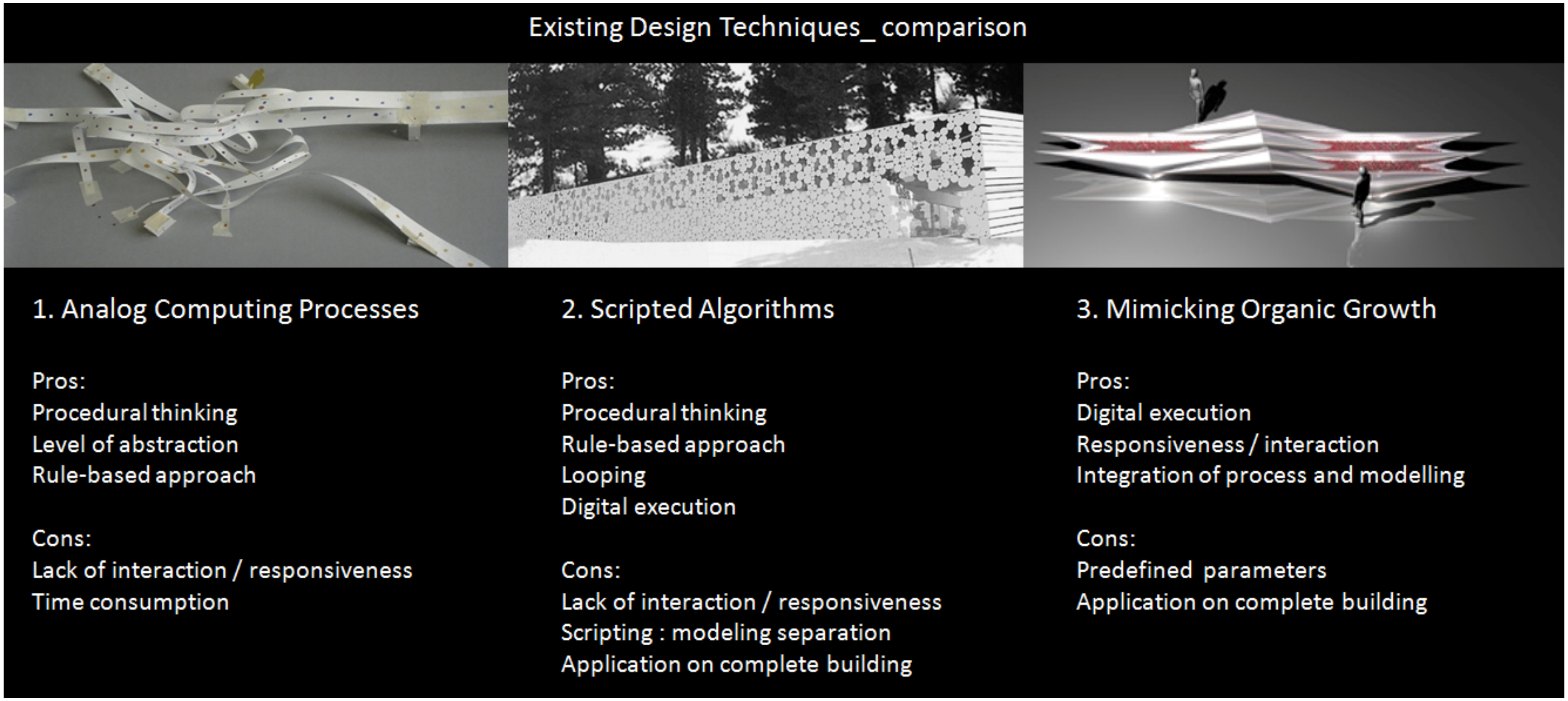
f. Undulation: This is a global measure of the variations in the Z plane. It controls the assembly of component population.

3. Changes in the fitness criteria dynamically change the population of the base component used to produce the bar's final configuration.

Comments:

This technique illustrates how a design system could be responsive to interruption and changes done by the designer, and how it could be open to adjustments in the parameters controlling its fitness criteria. The design's execution in a digital environment allows for fast and precise interactions between the designer and the resulting geometries. The strawberry bar project demonstrates a useful integration of design decisions and modeling of the involved geometries. However, the use of software-predefined parameters limits the designer's possibilities and constraint the selected processes of system growth.

Comparison of existing digital techniques:



Existing Design Techniques_ comparison

1. Analog Computing Processes

Pros:
Procedural thinking
Level of abstraction
Rule-based approach

Cons:
Lack of interaction / responsiveness
Time consumption

2. Scripted Algorithms

Pros:
Procedural thinking
Rule-based approach
Looping
Digital execution

Cons:
Lack of interaction / responsiveness
Scripting : modeling separation
Application on complete building

3. Mimicking Organic Growth

Pros:
Digital execution
Responsiveness / interaction
Integration of process and modelling

Cons:
Predefined parameters
Application on complete building

2.6 Comparison of Existing Digital Techniques

Conclusion:

All the three identified techniques in digital design clearly illustrate how the rule-based logic is used in design and form generation. Analysing the case studies for these techniques signifies the importance of design decision abstraction to translate the design problem solving into a rule-based procedure. In this approach to design, the rule-setting process is a key to the success or failure of any specific design because it is the direct way for the designer to translate ideas for the system to process.

In Son-O-House, I find the translation of body movements into paper is overly abstracted, although this could be argued acceptable in the scope of a sculpture project.

All the above-mentioned techniques could serve the rule-based approach to design proposed in this thesis. However, there are a few cons to each of the techniques:

The analog computing process could easily be too tiring and time consuming to process, once the factors that influence design decisions get too complicated. The ability to handle several design decisions simultaneously is limited in this technique because of the generated complexity. Computational environments are needed for simultaneous processing.

Both the analog computing processes and the scripted algorithms techniques lack the responsiveness needed to alter and manipulate existing designs. In either techniques, the project has to progress on its preset path once the rules are set. There are no means of adapting to changes in rules or parameters. To modify the model, there is no other choice but to delete the model, and re-run the script with changes to the driving parameters or to the script code.

The third technique offers a positive aspect in this issue. Design and model responsiveness is made possible by integrating the constraints into the model in one process. Changes made to any of the constraints causes a dynamic change in the entire model.

The scripted algorithms technique is very useful in the integration of loops, which are a sequences of statements that are specified once, but carried out a number of times or until specific conditions are met. This attribute allows for creating populations of a design typology, with slight variations occurring due to changing variables.

Breeding a population of designs can also be executed using the third technique of mimicking organic growth, but in this case, the population is limited to the predefined parameters built into the program. The scripted algorithms technique holds more potential in this area of design.

A significant issue is the limitation of design problems. The ability of the studied techniques to tackle complete designs is still limited and questionable. The analysis of the design problems illustrated in the case studies reveals a dominant interest in generating formal complexity without much concern for context, program, or function. To date, the drivers that influence form-making decisions in digital designs are often too abstracted to address programmatic design issues. A bigger challenge is in designing spaces considering issues related to user functions and project programs, compared to producing complex open space structures and surface patterns or geometries. User-oriented programmatic functions hold many limitations for the production of form; which, if accompanied by the still limited degree of control associated with algorithmic techniques, explains the disregard of program-sensitive design approaches in the field.

A key contribution of this thesis is the attempt to use context, project program, and user-sensitive design drivers in a digital design process to respond to a larger framework of references.

Parametric Design:

In order to apply the context and program-sensitive design approach in a computational rule-based setting, a technique to coordinate and control multiple processes in a responsive and operable environment is essential. Parametric design techniques have the advantages of scripting techniques, only in a more interactive atmosphere. Therefore, by using the parametric, designers have a dynamic interactive relationship with the components of their designs. A designer is able to manipulate, control, interrupt, and interact with a visual process of design.

Because scripting is consistent in the parametric model, the parametric design technique provides the designer with more control to manipulate the design system at any point of time without having to re-run scripts. The designer's own scripts can define the geometric behavior of components even when being manipulated in dynamics. Any parametric and logical dependencies established between the model components are persistent throughout the model and can be maintained during interactive modification.

Parametric modeling incorporates the design decisions and constraints set by the designer into the design and modeling process. As in the case of model buildings in CAD, parametric modeling is centered on geometric features, but offers the ability of storing associative relationships between elements. In other words, parametric modeling precedes CAD design modeling in the explicit storing of design rules, so that we don’t have to. It entails a level of abstraction to reduce a design into a network of interrelated geometries. This might require reducing walls to lines and buildings to simple geometries. However simple the system may seem, like the control panel, it holds control over reading and manipulating the concrete object it represents.

In order to construct a functional responsive design system, a higher level of abstraction is needed in the parametric set to reduce a design problem to its core factors, again much like how control panels operate. Parametric modeling uses logic, expressions, and numbers to manipulate complicated design systems. In its simplest task, ‘*Revit*’’[5], a parametric modeller widely used in architecture offices, smartly recalculates a building's form each time the architect manipulates its floor heights in numbers. This responsive software is pre-programmed to understand certain number manipulations and translate them into graphical representation of architecture, taking into consideration all the interacting relationships between floors, slabs, heights, and the like. However, *‘Revit’* is targeted towards the mainstream architecture offices and is missing a fundamental aspect much needed to open possibilities for designers: This is the ability for end-used programming by allowing users to define their own design elements and extend the system to fit their various design needs.

Opposed to a production tool, *‘GenerativeComponents’*[6] is a parametric modeling software that allows for this flexibility in the design exploration stage. Once the underlying logic and design relationships have been defined, the designer can create new options and features that extend the system beyond its pre-programmed features. This has a great effect on exploring design by encouraging an iterative search for more efficient solutions. However, *‘GenerativeComponents’* is not an architectural production tool: it lacks predefined architecture-specific objects and the ability to document full architectural projects.

"A key concept of GenerativeComponents is the ability to define geometry in different ways. One approach is based on traditional CAD modeling expanded by parametric associative aspects. A second approach is based on writing simple scripts that generate geometry and associations. A third one is through programming in the Microsoft Visual Studio environment". (Aish)

What I find most powerful about this program is that it is possible to integrate these three approaches in parallel and apply them at different times towards the same system, while sharing the same design components. Non-programmers are given the opportunity to explore operations that would only be possible in scripting and computer programming. A designer-defined assembly of pre-programmed features can be captured and turned into new features for later use. This function does not require users to have programming expertise.

Opposed to a CAD modeling program, the investment is not so much in the geometries created, but in the relationships and dependencies that define how the geometries interact and respond to their references.

In the next chapters, parametric modeling in GenerativeComponents is used in a series of design explorations to define and control dependencies between a model and its programmatic, social, and environmental influences. The digital model is used to store explicit performance-sensitive design decisions and constraints in a setting that enables dynamic modification and response. The design explorations challenge the capacity of parametric design techniques to manipulate not only form, but also function, program and space.

[4] Michael Hensel, Achim Menges, Michael Weinstock. "Frei Otto in Conversation with the Emergence and Design Group." Architectural Design 2004: 18-25.

[5] A building information modeling software by Autodesk.

[6] A parametric modeling software by Bentley.

3 Space Configurations

Introduction

The first exploration applies parametric and computational techniques in design to generate different spatial configurations inside typical residential units. The exploration works at the small scale of basic living units in a dense city core, where residential towers offer the most compressed domestic spaces. It explores spatial studies on a unit inside a typical residential tower, with a goal to open a wide range of possibilities for occupation.

Design Approach

Living in dense cities is the context for this exploration. Social factors like lifestyle and living scenarios of potential occupants together with environmental factors like location, orientation and view are the drivers for occupation change. In this exploration, these factors not only define the performance aspects of user occupation, but also the spatial configurations that accommodate them. Parametric techniques and computation are used to design for changing occupation by modeling both the occupation constraints and the design components into one system. This technique offers an alternative mode of abstraction to a conventional design technique, and captures portions of the mind design processes for the designer to analyse in a visual environment.

Space

Space, in the explored small high-rise living units, is defined by its boundaries, physical, and non-physical separations. Physical separations are the hard interfaces inside a given space; walls, partitions, floors and skin. While non-physical separations can be performed by lighting, sound, color, ownership, etc. This exploration works within the existing fixed box that defines the boundaries of a compressed condominium unit. The assumption of fixed outer walls and ceiling limits the exploration to the inside of the living space without affecting the overall tower structure. The exploration focuses on the physical separations of space and the iterations of its components to generate alternative living environments.

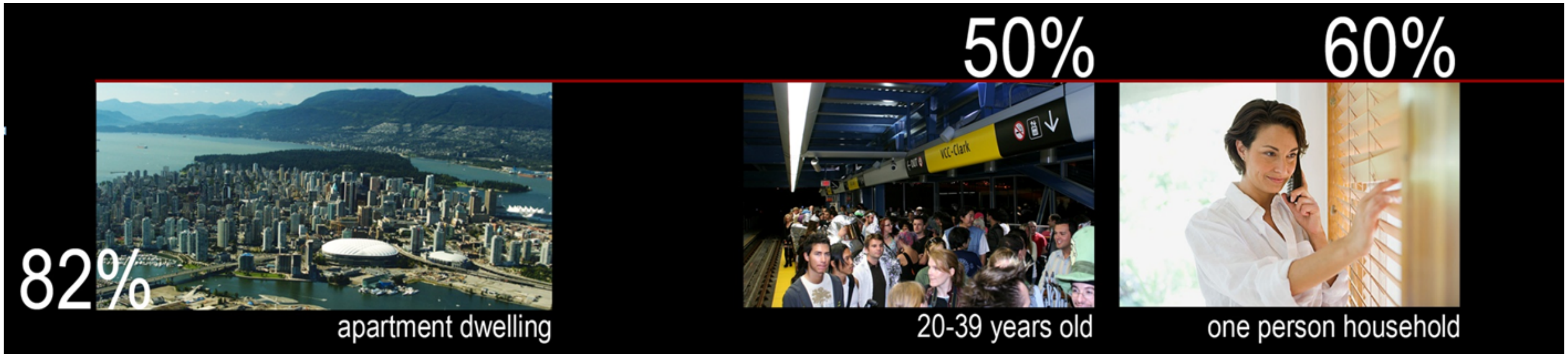
Occupation

Occupation, or Living in spaces, can be defined by the activities performed within the space. Working, exercising, eating, entertaining and sleeping all require different spatial qualities of space. In a typical high-rise housing scenario, users are normally classified into categories according to the number of household members. They typically choose between the units marketed for their usage category. Users often have to adapt to space in a way that best fits their occupation needs.

This method of categorizations tells very little about the actual occupation of space. Potential users have different interests and needs that has to be explored and designed for; which can dramatically change the inside configuration of their living units. This exploration aims at using computational design techniques to let the changing aspects of user occupation define the configuration of space, and not the opposite.

Existing Social Condition

The existing dwelling conditions in downtown Vancouver suggests the majority of living units fall under the one-bedroom type category. According to the 2001 census[7], 82.6% of Vancouver's downtown dwellings are apartment units in high-rise (5+ storeys) buildings. This number highlights the residential density in Vancouver's city core, which has only been intensified with office towers often converting to residential ones in recent years.



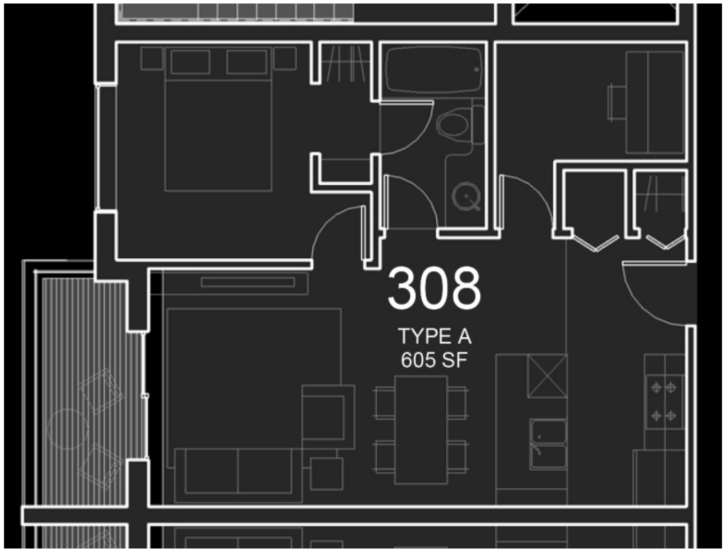
3.1 Existing Social Condition in downtown Vancouver

Half the downtown population falls in the 20-39 age group, with the number of one person households reaching 60.1% of all households. (Figure 3.1)

Typical Unit

Categorization: The residential towers in Vancouver are home to a variety of studios, one, two and three bedroom units, as classified by both design and marketing. The above numbers together with a look at the existing housing market in downtown Vancouver confirms that the vast majority of living units are one-bedrooms.

Open Concepts: The analysis of downtown Vancouver's typical unit layouts demonstrates the integration of open kitchens into living spaces to give the perception of spacious living and offers more interactions within the space.



3.2 Typical Living Unit

Efficiency: Space efficiency is a major requirement; entrance lobbies and hallways are diminishing, bathrooms in most cases are directly connected to the living area and often they have another access to the bedroom. Bedrooms are directly accessed from the living area. (Figure 3.2)

View Exposure: A great emphasis is put on the living space and bedroom's exposure to mountain and city views.

Position

Categorization and the lack of variation in the living units produce a highly standardised process of designing unit layouts. This approach forces the potential users to adapt to the preconfigured spaces existing in a given category. The actual method of occupation in such spaces varies from one user to the other, and this should potentially change the inside configuration of an apartment. I believe the designed living spaces have to offer an array of conditions that fit the array of possible users.

Proposal

This exploration uses computational design techniques to generate different spatial configurations inside typical domestic spaces, driven by the social and environmental context.

Design system

A parametric design system is set to regulate the relationship between the sub-spaces inside the unit, and control the physical separations that define the space. In this system, alterations at the level of a component affect other components of the system in a generative and cumulative behaviour.

Parametric modeling in *GenerativeComponents* is used to create the ‘Space Configurations’ design system. Rule-based procedures are used to create a responsive model-scale design system. The relationships between the design components are defined, and the rules are set to regulate their interactions. Constraints are also modeled together with the system's components to shape how the system works.

Objective

1. The main goal for this design system is to find an array of possible space configurations that respond to a variety of influencing social and environmental criteria.

2. This system should have the potential to act as a mass customisation tool for apartment models that controls and assembles the usual model components in different ways to produce different configurations for potential users.

3. The system should also provide a dynamic base for interior designers to start from to model new possibilities for occupation.

3.3 Components of Living Unit

Design Process

Components of model unit: The design process of the parametric system starts by analysing the subspaces that form together the typical living unit:

1. Living area

2. Bedroom

3. Kitchen

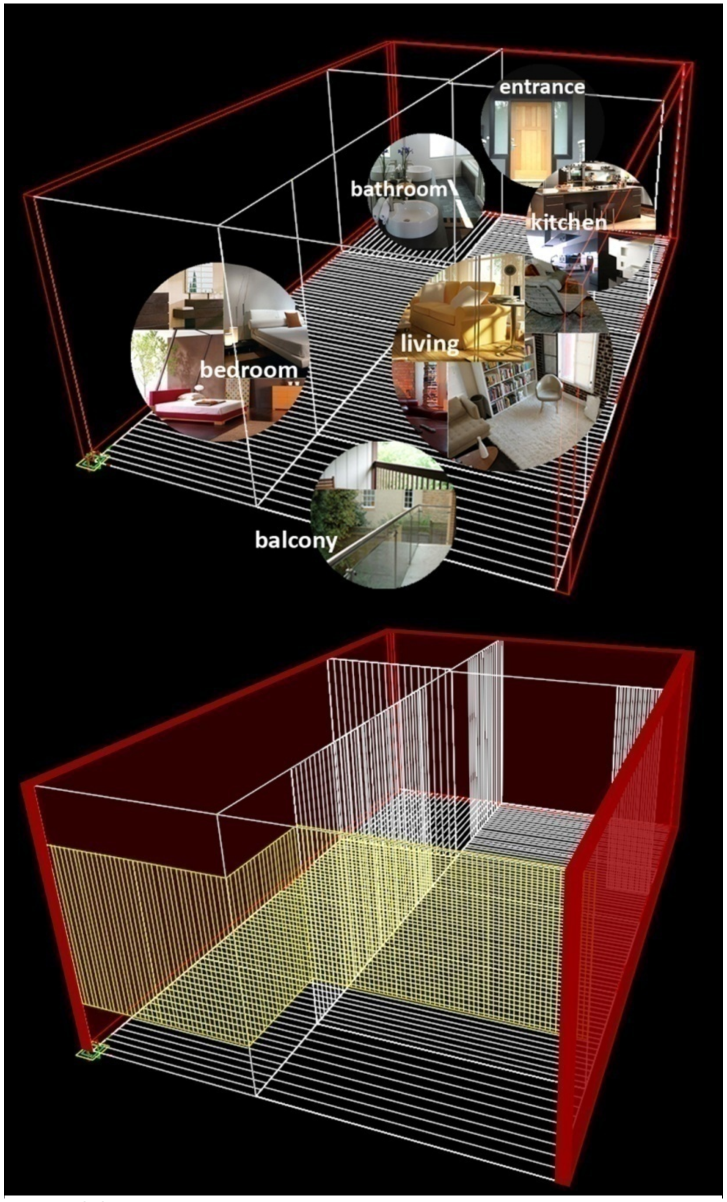
4. Bathroom

5. Balcony

6. Entrance

Relationship between components:

The analysis typical floor plan layouts show the relationships between the subspaces inside a living unit. Kitchens directly open to living rooms, entrance lobbies often do not exist in the compact size units or are placed within the open kitchen space. Bathrooms directly connect with both the living room and the bedroom. Bedrooms are directly accessed from the main living space. Balconies are placed in close proximity to the living space. Exposure to mountain and city views is encouraged in living rooms and bedrooms if possible. These relationships will be used in the parametric model to define and constrain the location of sub-spaces and the model's components.



3.4 Model Unit

Parametric Model Unit

The typical living unit is modeled in *GenerativeComponents* to define and regulate the relationship between the inside subspaces. The model components are the geometries that define the subspaces, like walls, floors, ceilings, etc. If we consider a residential floor is composed of an array of this base unit:

The red boundary wall is fixed, the position of interior walls and floors in whites are controlled parametrically, so is the glazing surface in yellow.

Defining Variables

1. Floors:

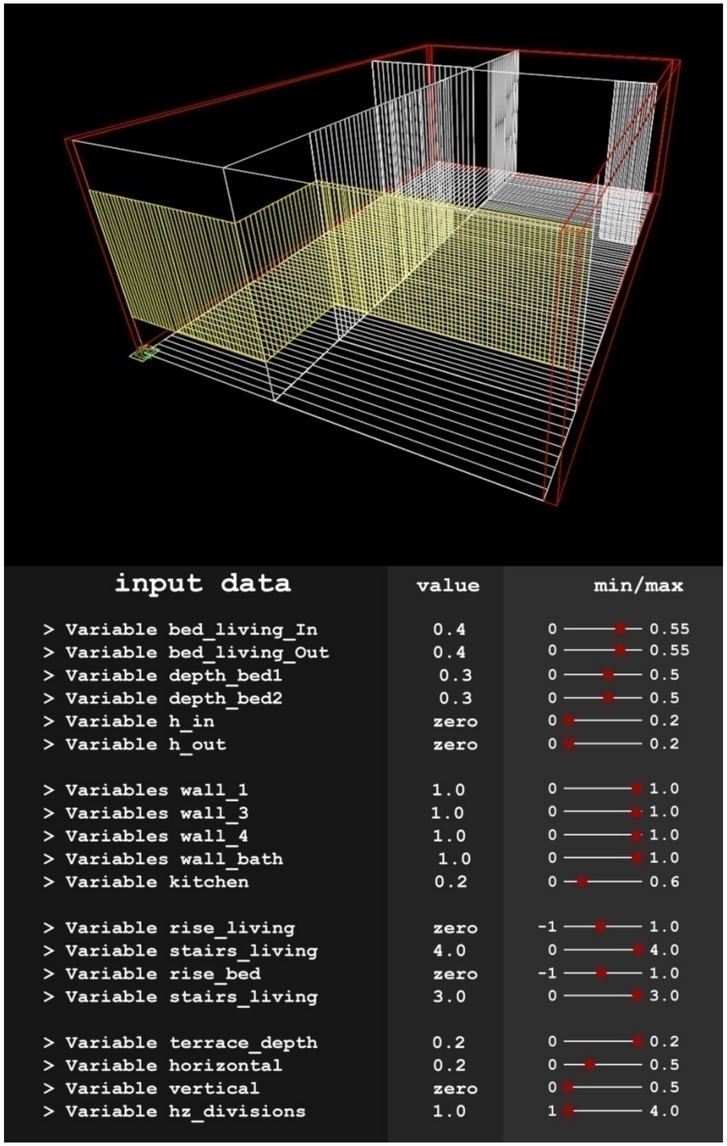
Each of the floor components inside the model unit can either be:

I. Flat   II. Stepped platforms   III. Ramped

For the floors to accommodate possible variations, variables are introduced to manipulate the heights at the each of the floor edges, the number of platforms as well as the height difference between platforms (value: 0 if flat), and the size of each floor component that affect room sizes.

2. Walls:

Vertical separations between the unit's components are achieved by partitions, which can be controlled by variables to be either:



input data	value	min/max
> Variable bed_living_In	0.4	0 —●— 0.55
> Variable bed_living_Out	0.4	0 —●— 0.55
> Variable depth_bed1	0.3	0 —●— 0.5
> Variable depth_bed2	0.3	0 —●— 0.5
> Variable h_in	zero	0 ●— 0.2
> Variable h_out	zero	0 ●— 0.2
> Variables wall_1	1.0	0 —● 1.0
> Variables wall_3	1.0	0 —● 1.0
> Variables wall_4	1.0	0 —● 1.0
> Variables wall_bath	1.0	0 —● 1.0
> Variable kitchen	0.2	0 —●— 0.6
> Variable rise_living	zero	-1 —●— 1.0
> Variable stairs_living	4.0	0 —● 4.0
> Variable rise_bed	zero	-1 —●— 1.0
> Variable stairs_living	3.0	0 —● 3.0
> Variable terrace_depth	0.2	0 —● 0.2
> Variable horizontal	0.2	0 —●— 0.5
> Variable vertical	zero	0 ●— 0.5
> Variable hz_divisions	1.0	1 ●— 4.0

3.5 **Model Variables**

I. Full height II. Percentage of full height III. Non-existing

3. Skin:

The exterior side of the unit is covered with a skin system, which is manipulated to adjust the size and ratio of solid walls to glazing. Variables control the horizontal and vertical size openings within this exterior system.

Variables Limits

Figure 3.6 shows the variables used for controlling the floors, walls, and skin components of the parametric model unit. The value numbers shown in the figure describe the initial configuration of the model. Upper and lower limits for these variables are set to control the desired range of configuration change for each of the components. For example, variables ‘bed_living_In’ and ‘bed_living_Out’ control the inner and outer floor separation between the living room and the bedroom. These two variables also control the position of the partition walls that separate the two spaces, controlled in height by variables: ‘wall_1’, ‘wall_2’, ‘wall_3’.

The range of set values for the variables represents the extreme configuration conditions. Beyond these values, the system’s configurations are not valid.

Defining Constraints:

Design constraints, which normally only exist in the designer's mind, are in this case computationally embedded into the design model. This allows for constant regulation of the design model behaviour and the visual display of the process for the designer. These constraints are also considered rules under which the system's variables control the interaction between components, and therefore decide how the system performs.

For this exploration, the constraints are defined after considering the relationships between the unit's components and the possible variation of form. These constraints also reflect on the range of set values for the model variables:

1. Side and inner walls are fixed.

2. Same level access between the main living space, bedroom, and bathroom.

3. Living space and bedroom have outside exposure.

4. Balcony has direct access to living space.

5. Bedroom and bathroom have a common wall.

6. If ramps are included, floors do not exceed 1:20 inclination.

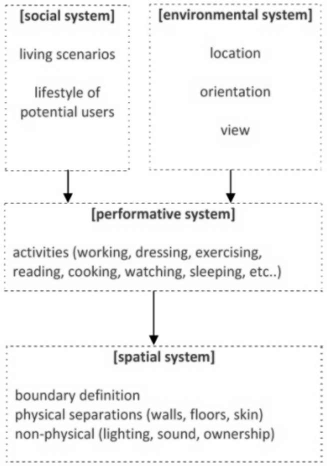
7. Height difference between platforms do not exceed 1ft.

8. Minimum ceiling height allowed is 6 ft.

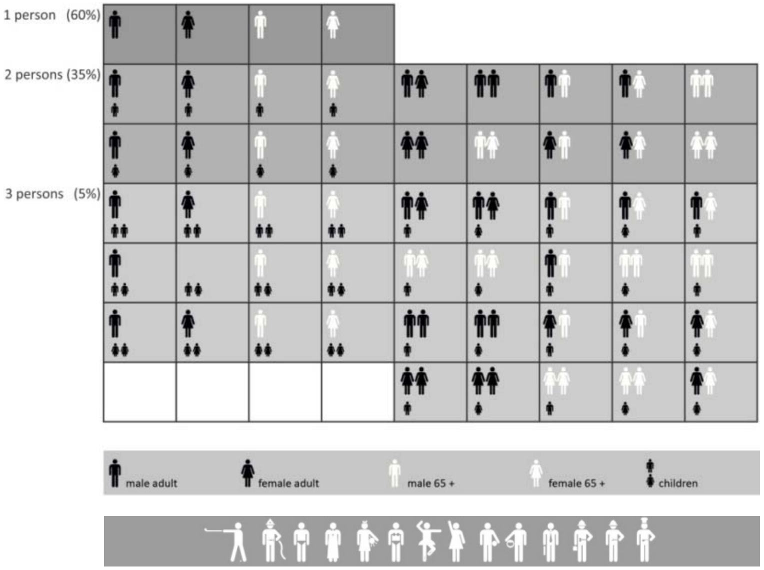
9. Minimum side dimension for bathroom: 20% of the unit depth.

10. Minimum width for living space: 30% of the unit width.

11: Minimum width for bedroom: 30% of unit width.


[social system]
living scenarios
lifestyle of potential users
[environmental system]
location
orientation
view
[performative system]
activities (working, dressing, exercising, reading, cooking, watching, sleeping, etc..)
[spatial system]
boundary definition
physical separations (walls, floors, skin)
non-physical (lighting, sound, ownership)

3.6 Design Initiation



3.7 Possible Households

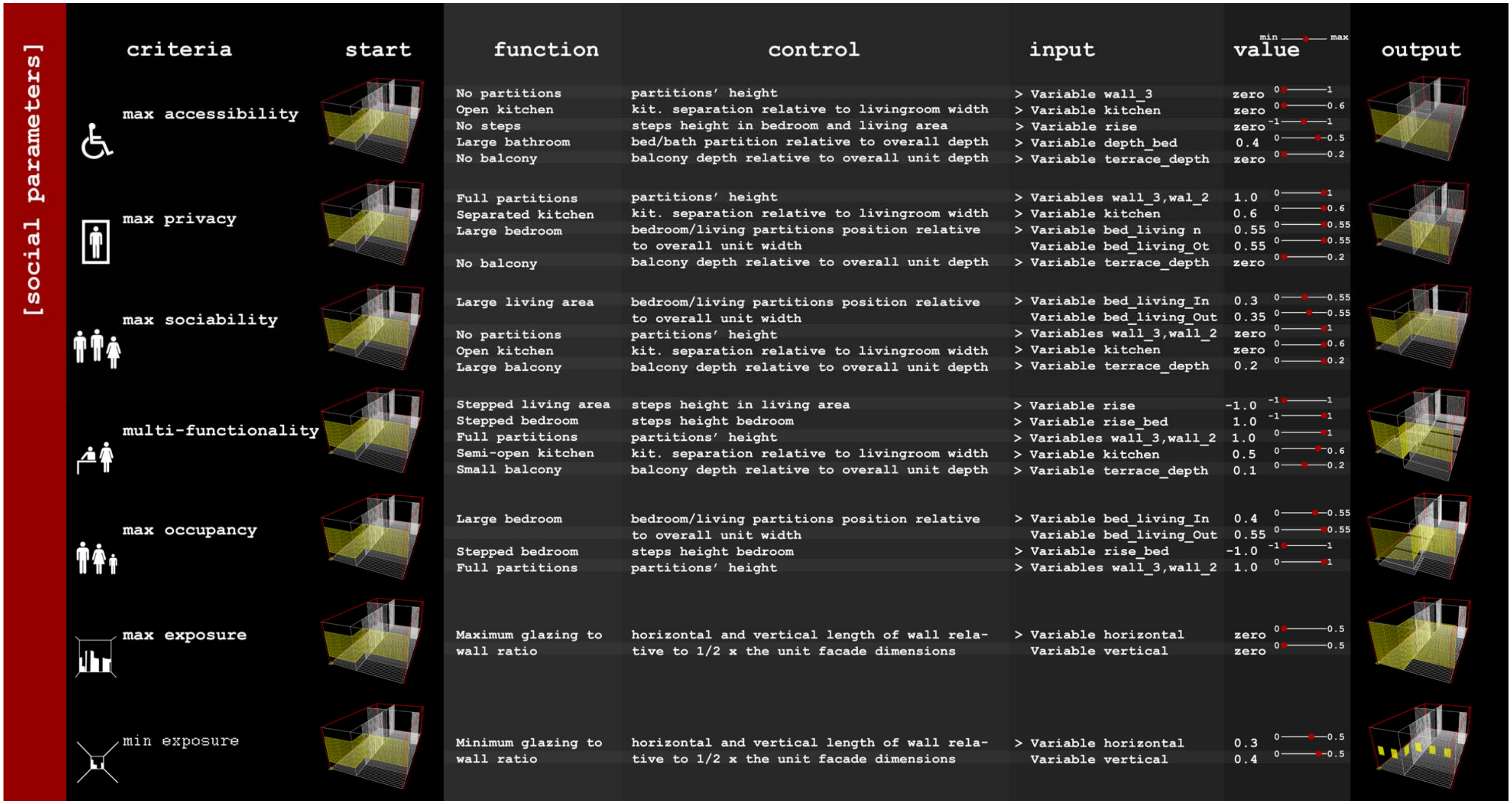
Design initiation

Both social and environmental factors are the drivers for the system's alterations. The performance of space users are driven by both the social and the environmental system they exist in. The social system is represented in the occupants' number, needs, lifestyle, and living preferences. The environmental system is represented by the unit's floor location inside the building, its orientation and cardinal direction, and its exposure to view. In this exploration, all the above factors not only define the performative aspects of user occupation, but also the spatial configurations that accommodate them. (Figure 3.7)

Rules are set inside the parametric design system to control the generation of spatial configurations. A specific combination of social and environmental factors triggers a corresponding change in the system's variables. The variables that control the system's components are manually controlled by the designer and the system dynamically responds to the designer's manipulations.

Social system

The analysis of census statistics allows for the assumptions of possible households scenarios living in one-bedroom apartments. Figure 3.8 describes possible number, gender and age of members in one-bedroom households. Taking into consideration the lifestyle difference between members of each group gives a greater range of living possibilities.



[social parameters]

criteria	start	function	control	input	value	min	max	output
max accessibility		No partitions	partitions’ height	> Variable wall_3	zero	0	1	
		Open kitchen	kit. separation relative to livingroom width	> Variable kitchen	zero	0	0.6	
		No steps	steps height in bedroom and living area	> Variable rise	zero	-1	1	
		Large bathroom	bed/bath partition relative to overall depth	> Variable depth_bed	0.4	0	0.5	
		No balcony	balcony depth relative to overall unit depth	> Variable terrace_depth	zero	0	0.2	
max privacy		Full partitions	partitions’ height	> Variables wall_3,wal_2	1.0	0	1	
		Separated kitchen	kit. separation relative to livingroom width	> Variable kitchen	0.6	0	0.6	
		Large bedroom	bedroom/living partitions position relative to overall unit width	> Variable bed_living n	0.55	0	0.55	
				Variable bed_living_Ot	0.55	0	0.55	
		No balcony	balcony depth relative to overall unit depth	> Variable terrace_depth	zero	0	0.2	
max sociability		Large living area	bedroom/living partitions position relative to overall unit width	> Variable bed_living_In	0.3	0	0.55	
				Variable bed_living_Out	0.35	0	0.55	
		No partitions	partitions’ height	> Variables wall_3,wall_2	zero	0	1	
		Open kitchen	kit. separation relative to livingroom width	> Variable kitchen	zero	0	0.6	
		Large balcony	balcony depth relative to overall unit depth	> Variable terrace_depth	0.2	0	0.2	
multi-functionality		Stepped living area	steps height in living area	> Variable rise	-1.0	-1	1	
		Stepped bedroom	steps height bedroom	> Variable rise_bed	1.0	-1	1	
		Full partitions	partitions’ height	> Variables wall_3,wall_2	1.0	0	1	
		Semi-open kitchen	kit. separation relative to livingroom width	> Variable kitchen	0.5	0	0.6	
		Small balcony	balcony depth relative to overall unit depth	> Variable terrace_depth	0.1	0	0.2	
max occupancy		Large bedroom	bedroom/living partitions position relative to overall unit width	> Variable bed_living_In	0.4	0	0.55	
				Variable bed_living_Out	0.55	0	0.55	
		Stepped bedroom	steps height bedroom	> Variable rise_bed	-1.0	-1	1	
		Full partitions	partitions’ height	> Variables wall_3,wall_2	1.0	0	1	
max exposure		Maximum glazing to wall ratio	horizontal and vertical length of wall relative to 1/2 x the unit facade dimensions	> Variable horizontal	zero	0	0.5	
				Variable vertical	zero	0	0.5	
min exposure		Minimum glazing to wall ratio	horizontal and vertical length of wall relative to 1/2 x the unit facade dimensions	> Variable horizontal	0.3	0	0.5	
				Variable vertical	0.4	0	0.5	

3.8 Social System Parameters

A number of possible social influences that consider issues of accessibility, privacy, sociability, multi-functionality, occupancy, and desired level of exposure are identified into the schedule in figure 3.9. These influences address specific related functions that control the parametric model’s geometries and components. Each of the criteria is associated with related values for the component variables for the system to reach best fitting spatial configuration, shown in the output column of the schedule.

Environmental system

The environmental system mostly controls the glazing surfaces of the living unit. Changes are triggered by both the orientation of the unit and its floor location relative to the tower.

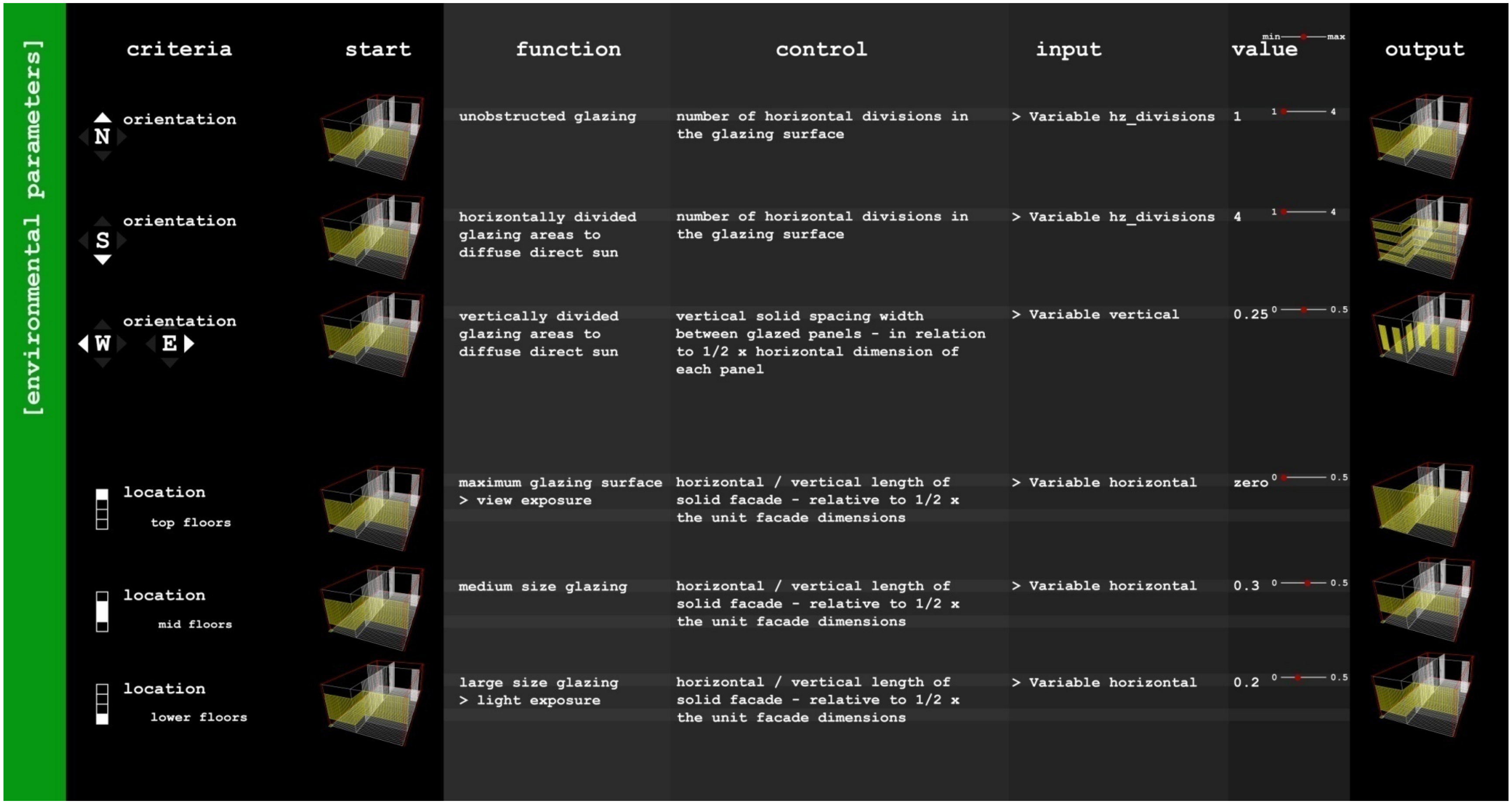
Orientation: There is a great potential for this model to accommodate horizontal and vertical sun shading devices. Horizontal glazing divisions eliminate direct sunrays and minimise the heat gain on south facing facades. Vertical divisions are also parametrically controlled to minimise problematic rays on west facades.

Floor location: Lower floors in high-rise buildings tend to be less exposed to direct and diffused light thus the need for larger glazing areas to compensate for the decrease in lighting levels. Upper levels are normally associated with open views; therefore, unobstructed glazing surfaces are used.

Skin glazing components are modeled over the unit’s facade at both the bedroom and living spaces. They abstract the relationship between solid and glazed surfaces on each particular unit's facade. They are controlled parametrically to extend in horizontal and vertical directions and satisfy the exact form and glazing ratio. The factors affected are:

1. Horizontal glazing size and spacing

2. Vertical glazing size and spacing



[environmental parameters]

criteria	start	function	control	input	value	output
N orientation		unobstructed glazing	number of horizontal divisions in the glazing surface	> Variable hz_divisions	1	
S orientation		horizontally divided glazing areas to diffuse direct sun	number of horizontal divisions in the glazing surface	> Variable hz_divisions	4	
W E orientation		vertically divided glazing areas to diffuse direct sun	vertical solid spacing width between glazed panels - in relation to 1/2 x horizontal dimension of each panel	> Variable vertical	0.25	
location top floors		maximum glazing surface > view exposure	horizontal / vertical length of solid facade - relative to 1/2 x the unit facade dimensions	> Variable horizontal	zero	
location mid floors		medium size glazing	horizontal / vertical length of solid facade - relative to 1/2 x the unit facade dimensions	> Variable horizontal	0.3	
location lower floors		large size glazing > light exposure	horizontal / vertical length of solid facade - relative to 1/2 x the unit facade dimensions	> Variable horizontal	0.2	

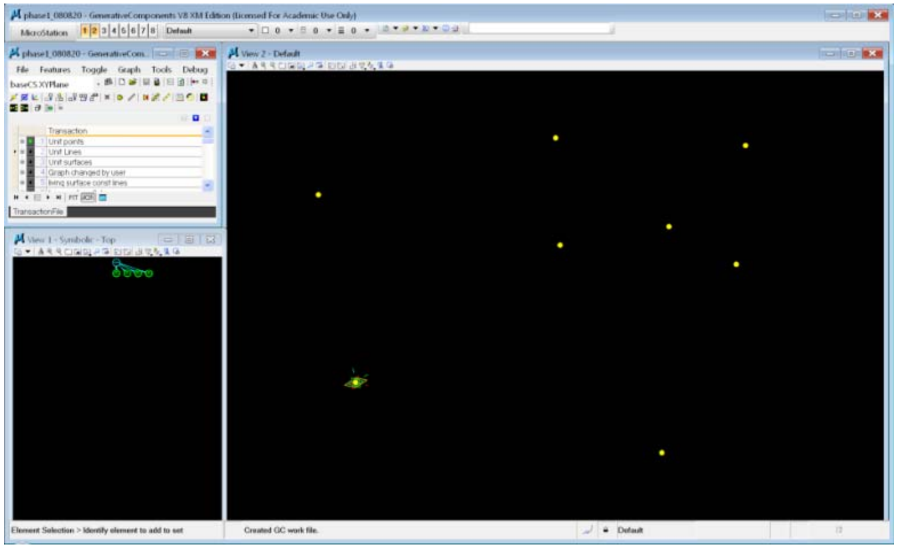
3.9 Environmental System Parameters

Figure 3.10 describes the environmental influences, functions, and control over the system’s glazing components.

```
transaction modelBased "Unit points"
{feature In_left GC.Point
    {
     CoordinateSystem          = baseCS;
        XTranslation              = 0;
        YTranslation              = 30;
        ZTranslation              = Series(0,12,12);
        HandlesVisible            = true;
    }
    feature In_right GC.Point
    {
        CoordinateSystem          = baseCS;
        XTranslation              = 20;
        YTranslation              = 30;
        ZTranslation              = Series(0,12,12);
        HandlesVisible            = true;
    }
    feature Out_left GC.Point
    {
        CoordinateSystem          = baseCS;
        XTranslation              = 0;
        YTranslation              = 0;
        ZTranslation              = Series(0,12,12);
        HandlesVisible            = true;
    }
    feature Out_right GC.Point
    {
        CoordinateSystem          = baseCS;
        XTranslation              = 20;
        YTranslation              = 0;
        ZTranslation              = Series(0,12,12);
        HandlesVisible            = true;
    }
}
```

System Making

The following describe the process of creating the parametric design system in *GenerativeComponents*. The model unit's components are modeled together with the rules, constraints, and variables that control the interaction between them. .



	Transaction
1	Unit points
2	Unit Lines
3	Unit surfaces
4	Graph changed by user
5	living surface const lines

Element Selection > Identify element to add to set

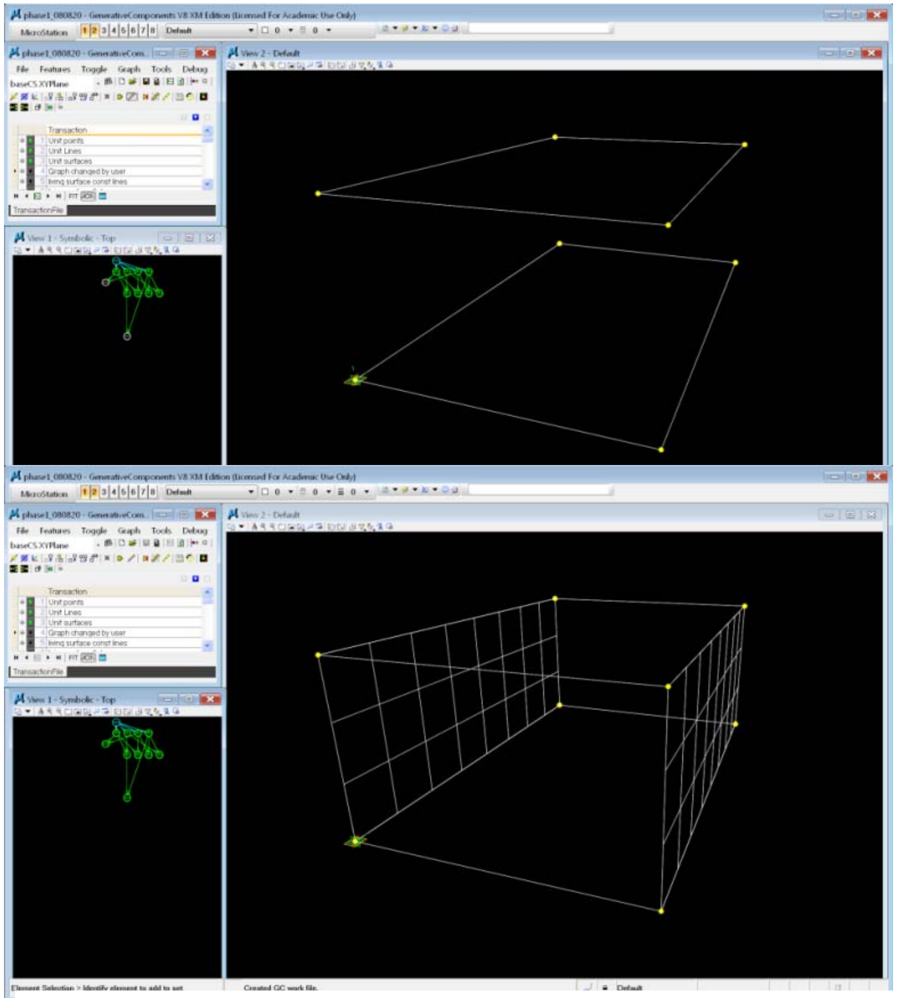
Created GC work file.

3.10

Unit points:

Corner points {In_left, In_right, Out_left, Out_right} created. Dimensions are 20ft W x 30ft D x 12ft H. Each of the corner points has a floor and ceiling instance: (Z Translation: Series(0,12,12)) where 12 is the height.

```
transaction modelBased "Unit Lines"
{
    feature line01 GC.Line
    {
        StartPoint                 = Out_left;
        EndPoint                   = In_right;
    }
}

transaction modelBased "Unit surfaces"
{
   feature line01 GC.Line
    {
        EndPoint                  = Out_right;
        SymbolXY                  = {103, 103};
    }
    feature line02 GC.Line
    {
        StartPoint                = In_left;
        EndPoint                  = In_right;
        SymbolXY                  = {100, 103};
    }
    feature line03 GC.Line
    {
        StartPoint                = In_left;
        EndPoint                  = Out_left;
        SymbolXY                  = {101, 103};
    }
    feature line04 GC.Line
    {
        StartPoint                = In_right;
        EndPoint                  = Out_right;
        SymbolXY                  = {102, 103};
    }
    feature surface_left GC.BSplineSurface
    {
        Points                    = {In_left,Out_left};
        UcurveDisplay             = 9;
        SymbolXY                  = {99, 102};
    }
    feature surface_right GC.BSplineSurface
    {   Points                    = {In_right,Out_right};
        UcurveDisplay             = 9;
    }
}
```

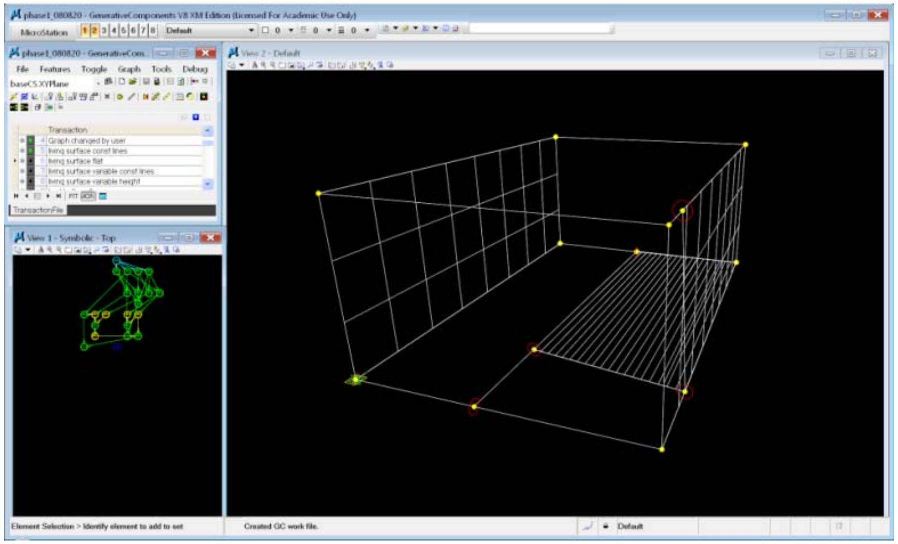


3.11

Unit surfaces:

Lines between each of the points created to define the boundary of the model unit. Surfaces {surface_left, surface_right} are ruled on the sides to define a fixed wall.

```
transaction modelBased "living surface const lines"
{
    feature In_mid GC.Point
    {
        Curve                     = line02[0];
        T                         = <free> (0.46101271829523);
        SymbolXY                  = {98, 105};
        HandlesVisible            = true;
    }
    feature Out_mid GC.Point
    {
        Curve                     = line01[0];
        T                         = <free> (0.437099879921155);
        SymbolXY                  = {99, 105};
        HandlesVisible            = true;
    }
    feature bsplineSurface02 GC.BSplineSurface
    {
        StartCurve                = line07;
        EndCurve                  = line08;
        UcurveDisplay             = 2;
        VcurveDisplay             = 20;
        SymbolXY                  = {97, 108};
    }

}
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
Transaction
TransactionFile
View 1 - Symbolic - Top
View 2 - Default

4 Graph changed by user
5 living surface const lines
6 living surface flat
7 living surface variable const lines
8 living surface variable height

Element Selection > Identify element to add to set

Created GC work file.

3.12

Living surface:

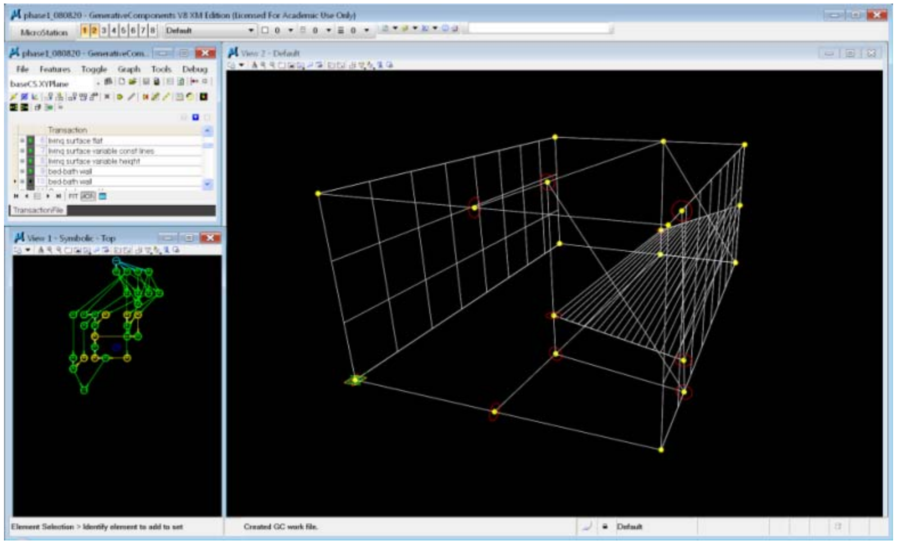
A line is constructed on the floor surface to divide the unit into 2 zones: a *kitchen + living + deck*, and a *bedroom + bathroom* area. Another line separates the living area from the deck. A surface (Surface02) now defines a flat projection of the living space.

Variable terrace_depth controls the deck's size.

Variable bed_living_In controls the inner ratio between the two zones.

Variable bed_living_Out controls the outer ratio between the two zones.

```
transaction  modelBased "living surface variable const lines"
{
    feature In_mid GC.Point
    {
        Curve                        = line02;
    }
    feature Out_mid GC.Point
    {
        Curve                        = line01;
        T                            = <free> (0.50688199162311);
    }
    feature line09 GC.Line
    {
        StartPoint                   = point05[0];
        EndPoint                     = point05[1];
        SymbolXY                     = {98, 108};
    }
    feature line10 GC.Line
    {
        StartPoint                   = point07;
        EndPoint                     = point06;
        SymbolXY                     = {99, 109};
    }
    feature line11 GC.Line
    {
        StartPoint                   = In_mid[0];
        EndPoint                     = In_mid[1];
        SymbolXY                     = {97, 106};
    }
    feature line12 GC.Line
    {
        StartPoint                   = In_right[0];
        EndPoint                     = In_right[1];
        SymbolXY                     = {96, 105};
    }
    feature line13 GC.Line
    {
        StartPoint                   = point09;
        EndPoint                     = point08;
        SymbolXY                     = {96, 110};
    }
    feature point06 GC.Point
    {
        Curve                        = line05;
        T                            = <free> (0.788988246693955);
        SymbolXY                     = {101, 109};
        HandlesVisible               = true;
    }
    feature point07 GC.Point
    {
        Curve                        = line09;
        T                            = <free> (0.244748397500457);
        SymbolXY                     = {98, 109};
        HandlesVisible               = true;
    }

    feature point08 GC.Point
    {
        Curve                        = line11;
        T                            = <free> (0.22809648460624);
        SymbolXY                     = {97, 109};
        HandlesVisible               = true;
    }
    feature point09 GC.Point
    {
        Curve                        = line12;
        T                            = .5058246;
        SymbolXY                     = {96, 109};
        HandlesVisible               = true;
    }
}

transaction modelBased "dynamic living surface"
{
    feature bsplineSurface03 GC.BSplineSurface
    {
        StartCurve                   = line10;
        EndCurve                     = line13;
        UcurveDisplay                = 2;
        VcurveDisplay                = 20;
        SymbolXY                     = {97, 112};
    }
    feature line13 GC.Line
    {
        StartPoint                   = point08;
        EndPoint                     = point09;
    }
}
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
Transaction
living surface flat
living surface variable const lines
living surface variable height
bed-bath wall
bed-bath wall
TransactionFile
View 1 - Symbolic - Top
View 2 - Default
Element Selection > Identify element to add to set
Created GC work file.
Default

3.13

Dynamic living surface:

A number of movable points and guide lines are constructed to provide a dynamic base for the living space floor. A floor surface (Surface03) is based on these dynamic points. The points are free to move on vertical guides, and therefore can change the floor's height at any of the corners.

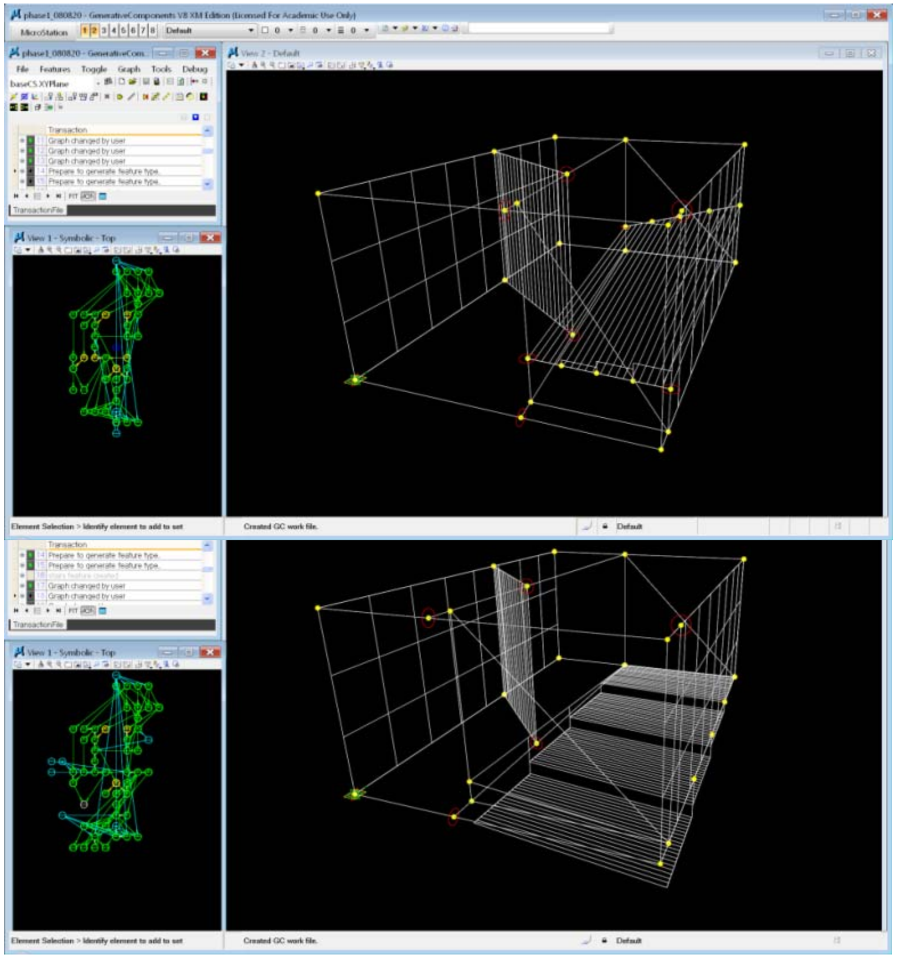
Bed-Bath transition:

Surface (Surface04)is ruled between guide lines that are free to move along the depth of the unit. It draws the boundary between the bedroom and bathroom in this unit.

*Variable h_In c*ontrols the inner floor spot elevation as percentage of full height.

Variable h_Out controls the outer floor spot elevation as percentage of full height.

```
feature rise GC.GraphVariable
    {
        Value                     = -0.26;
        RangeMinimum              = -1.0;
    }
feature stairs GC.GraphVariable
    {
        Value                     = 4.0;
        LimitValueToRange         = true;
        RangeMinimum              = 2.0;
        RangeMaximum              = 5.0;
        RangeStepSize             = 1.0;
        SymbolXY                  = {95, 113};


transaction generateFeatureType "stairs feature created",
suppressed
{
    type                        = GC.UBCStairs02;
    inputProperties             = {
                                        property GC.CoordinateSystem
baseCS
                                        {
                                            feature                   =
baseCS;
                                            isReplicatable            =
true;
                                            isParentModel             =
true;
                                        }
                                        property GC.Line line24
                                        {
                                            feature                   =
line24;
                                            isReplicatable            =
true;
                                        }
                                        property double rise
                                        {
                                            feature                   =
rise;
                                            isReplicatable            =
true;
                                        }
                                        property double stairs
                                        {
                                            feature                   =
stairs;
                                            isReplicatable            =
true;
                                        }
                                    };
    outputProperties            = {

                                        property GC.Line line18
                                        {
                                            feature                   =
line18;
                                            isDynamic                 =
true;
                                        }
                                        property GC.Line line20
                                        {
                                            feature                   =
line20;
                                            isDynamic                 =
true;
                                        }
                                        property GC.Line line21
                                        {
                                            feature                   =
line21;
                                            isDynamic                 =
true;
                                        }
                                        property GC.Line line22
                                        {
                                            feature                   =
line22;
                                            isDynamic                 =
true;
                                        }
                                        property GC.Point point12
                                        {
                                            feature                   =
point12;
                                            isConstruction            =
true;
                                            isDynamic                 =
true;
                                        }
                                    };
}
```


phase1_080820 - GenerativeComponents V8 XM Edition (licensed For Academic Use Only)
View 2 - Default
View 1 - Symbolic - Top
Transaction
Graph changed by user
Prepare to generate feature type.
Element Selection > Identify element to add to set
Created GC work file.
Default

3.14

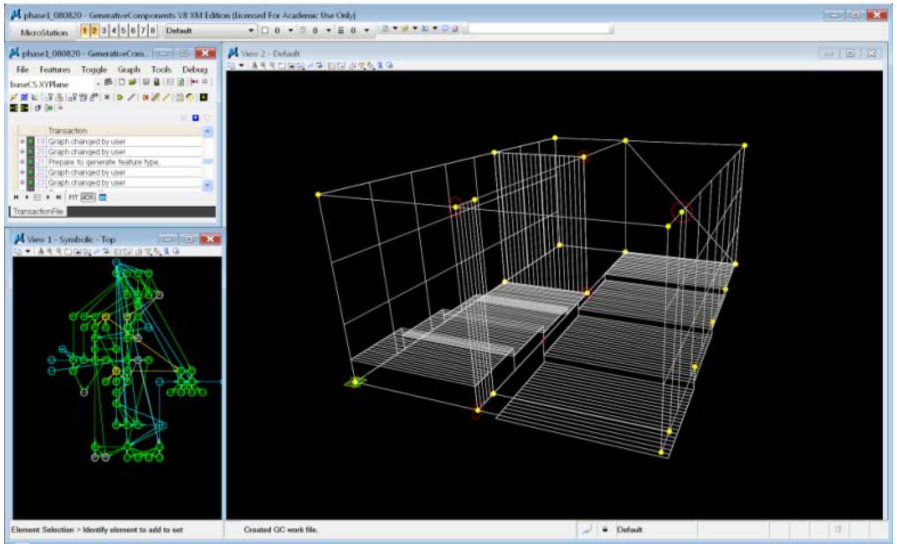
Platforms feature generated:

Platforms were created and applied to the living floor surface. Platforms are now an added feature to GC's user interface. Points, lines, directions, and surfaces are used to generate this feature.

Variable rise controls the height difference between platforms (-1ft to 1ft)

Variable stairs controls the number of platforms (2 to5)

```
feature rise_bed GC.GraphVariable
    {
        Value                     = .2;
        LimitValueToRange         = true;
        RangeMinimum              = -1.0;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {110, 111};
    }
    feature stairs GC.GraphVariable
    {
        SymbolXY                  = {104, 111};
    }
    feature ubcstairs0203 GC.UBCStairs02
    {
        baseCS                    = baseCS;
        line24                    = line29;
        rise                      = rise_bed;
        stairs                    = stairs;
        SymbolXY                  = {107, 111};
    }
    feature ubcstairs0204 GC.UBCStairs02
    {
        baseCS                    = baseCS;
        line24                    = line30;
        rise                      = rise_bed;
        stairs                    = stairs;
        SymbolXY                  = {106, 111};
    }
}
```



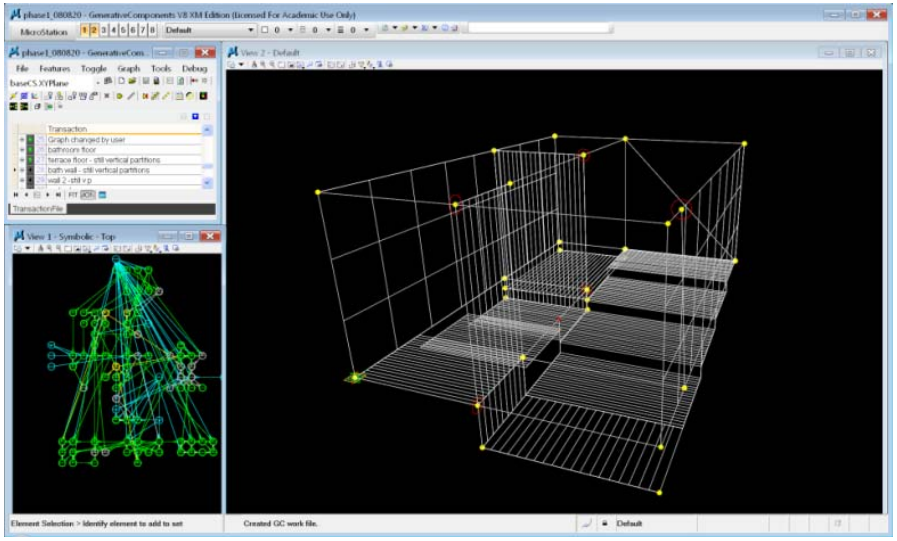
3.15

Bedroom platforms:

Platforms are constructed over the bedroom floor space. The bedroom entrance (on 1st bedroom platform) has the same height as the 3rd living space platform.

Variable rise_bed controls the height difference between bedroom platforms (-1ft to 1ft).

```
transaction modelBased "Bedroom platforms"
{
    feature bsplineSurface11 GC.BSplineSurface
    {
        StartCurve                = ubcstairs0203.line22;
        EndCurve                  = ubcstairs0204.line22;
        UcurveDisplay             = 2;
        VcurveDisplay             = 20;
        SymbolXY                  = {105, 112};
    }
    feature bsplineSurface12 GC.BSplineSurface
    {
        StartCurve                = ubcstairs0203.line21;
        EndCurve                  = ubcstairs0204.line21;
        UcurveDisplay             = 2;
        VcurveDisplay             = 20;
        SymbolXY                  = {108, 112};
    }
    feature bsplineSurface13 GC.BSplineSurface
    {
        StartCurve                = ubcstairs0203.line20;
        EndCurve                  = ubcstairs0204.line20;
        UcurveDisplay             = 2;
        VcurveDisplay             = 20;
        SymbolXY                  = {106, 112};
    }
    feature bsplineSurface14 GC.BSplineSurface
    {
        StartCurve                = ubcstairs0203.line25;
        EndCurve                  = ubcstairs0204.line25;
        UcurveDisplay             = 2;
        VcurveDisplay             = 20;
        SymbolXY                  = {107, 112};
    }
    feature rise_bed GC.GraphVariable
    {
        Value                     = 0.54;
    }
}
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
View 2 - Default
View 1 - Symbolic - Top
TransactionFile

Transaction
25 Graph changed by user
26 bathroom floor
27 terrace floor - still vertical partitions
28 bath wall - still vertical partitions
29 wall 2 - still v p

Element Selection > Identify element to add to set

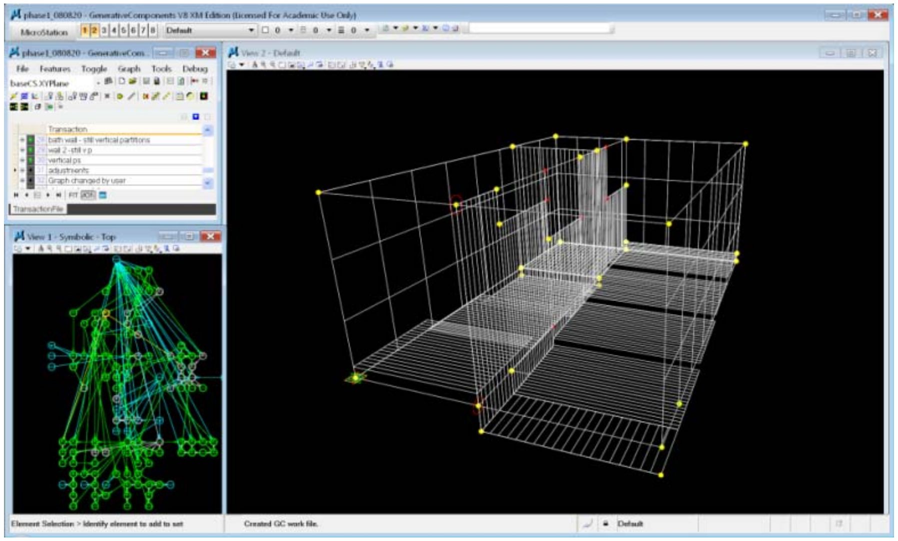
Created GC work file.

3.16

Floors:

The deck and bathroom floors are constructed. The deck shares the same floor height with the 4th platform on the living space, and the bathroom has the same height as the 2nd platform on the living space. These relations were decided according to the model unit's space layout and the relationships between its components.

```
feature wall_bath GC.GraphVariable
    {
        Value                     = 1.0;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.2;
    }

    feature wall_2 GC.GraphVariable
    {
        Value                     = 1.0;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
    }
    feature wall_bath GC.GraphVariable
    {
        Value                     = 0.6;
    }

transaction modelBased "vertical ps"
{


    feature wall_3 GC.GraphVariable
    {
        Value                     = 0.62;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {97, 121};
    }
    feature wall_4 GC.GraphVariable
    {
        Value                     = 0.85;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {95, 121};
    }
    feature wall_bath GC.GraphVariable
    {
        SymbolXY                  = {99, 121};
    }
}
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
Transaction
bath wall - still vertical partitions
wall 2 - still v p
vertical ps
adjustments
Graph changed by user
TransactionFile
View 1 - Symbolic - Top
View 2 - Default
Element Selection > Identify element to add to set
Created GC work file.
Default

3.17

Vertical separation:

Partitions are constructed along the floor line that divides the *kitchen+living+deck*, and the *bedroom+bathroom* areas. These are vertically extended from the living space platforms and are controlled by height variables each, to a percentage of the full height.

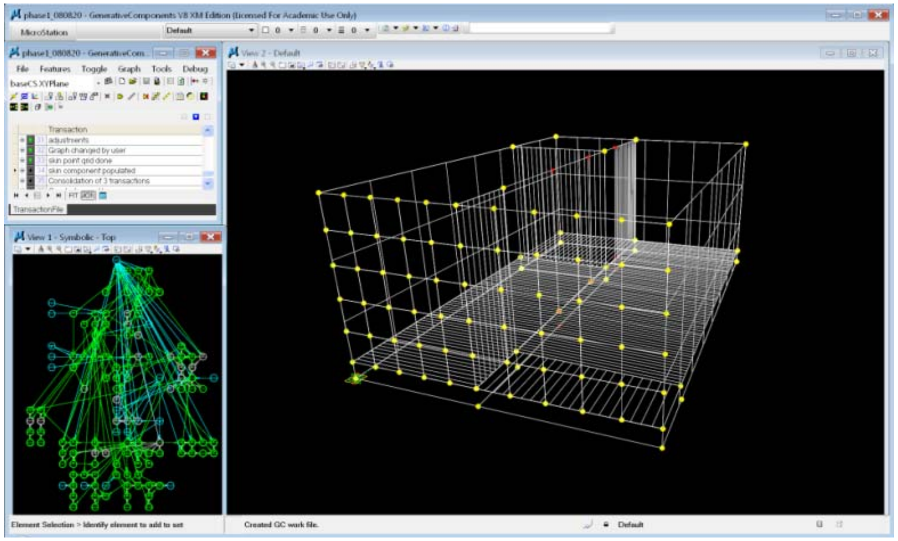
Variable wall_bath controls the 1st partition's height.

Variable wall_2 controls the 2nd partition's height.

Variable wall_3 controls the 3rd partition's height.

Variable wall_4 controls the 4th partition's height.

```
transaction modelBased "skin point grid done"
{
    feature bsplineSurface22 GC.BSplineSurface
    {
        StartCurve                 = line47;
        EndCurve                   = line26;
        SymbolXY                   = {93, 115};
        Visible                    = false;
    }
    feature bsplineSurface23 GC.BSplineSurface
    {
        StartCurve                 = line26;
        EndCurve                   = line46;
        SymbolXY                   = {92, 115};
        Visible                    = false;
    }
       feature point37 GC.Point
    {
        Surface                    = bsplineSurface23;
        U                          = Series(0,1,0.2);
        V                          = Series(0,1,0.2);
        SymbolXY                   = {92, 116};
        HandlesVisible             = true;
        Replication                =
ReplicationOption.AllCombinations;
    }
    feature point38 GC.Point
    {
        Surface                    = bsplineSurface22;
        U                          = Series(0,1,0.2);
        V                          = Series(0,1,0.2);
        SymbolXY                   = {93, 116};
        Replication                =
ReplicationOption.AllCombinations;
    }
    feature polygon01 GC.Polygon
    {
        Points                     = point37;
        SymbolXY                   = {92, 117};
    }
    feature polygon02 GC.Polygon
    {
        Points                     = point38;
        SymbolXY                   = {93, 117};
    }
}
```



Transaction
31 adjustments
32 Graph changed by user
33 skin point grid done
34 skin component populated
35 Consolidation of 3 transactions

Element Selection > Identify element to add to set

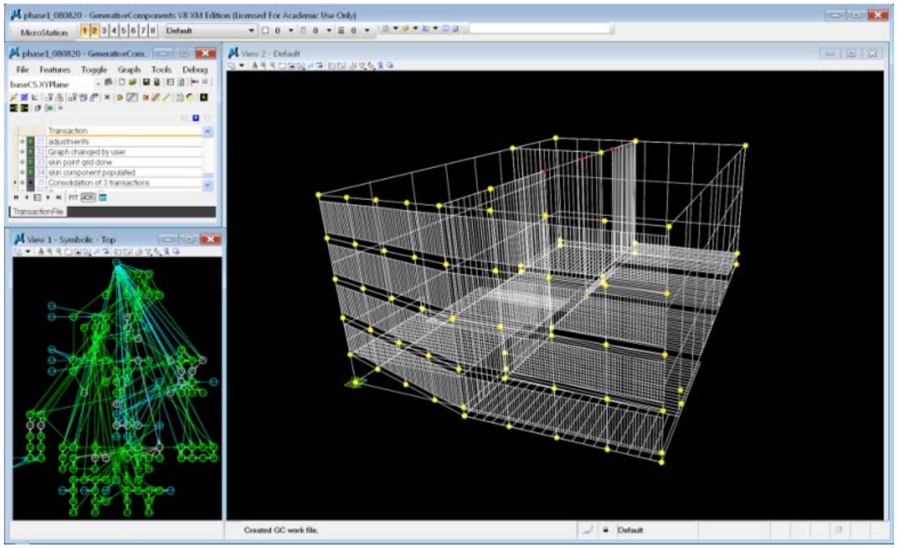
Created GC work file.

3.18

Skin point grid:

A surface mesh is constructed over the exterior side of the model unit. Points are populated on the surface to form a 5 x 10 grid (numbers adjustable).

```
transaction modelBased "skin component populated"
{
   feature horizontal GC.GraphVariable
    {
        Value                     = 0.14;
        LimitValueToRange         = true;
        RangeMaximum              = 0.5;
        RangeStepSize             = 0.0;
        SymbolXY                  = {90, 118};
    }
    feature line48 GC.Line
    {
        StartPoint                =
ubcstairs0201.line22.StartPoint;
        EndPoint                  = point31;
        SymbolXY                  = {99, 120};
    }
    feature point37 GC.Point
    {
        U                         = Series(0,1,0.25);
        V                         = Series(0,1,0.25);
        Visible                   = false;
    }
    feature point38 GC.Point
    {
        U                         = Series(0,1,0.25);
        V                         = Series(0,1,0.25);
        Visible                   = false;
    }
    feature ubcskinComponent0801 GC.UBCSkinComponent08
    {
        horizontal                = horizontal;
        polygon01                 = polygon01;
        vertical                  = vertical;
        SymbolXY                  = {92, 118};
    }
    feature ubcskinComponent0802 GC.UBCSkinComponent08
    {
        horizontal                = horizontal;
        polygon01                 = polygon02;
        vertical                  = vertical;
        SymbolXY                  = {93, 118};
        Visible                   = true;
    }
    feature vertical GC.GraphVariable
    {
        Value                     = 0.0;
        LimitValueToRange         = true;
        RangeMaximum              = 0.5;
        RangeStepSize             = 0.0;
        SymbolXY                  = {90, 119};
    }
}
```



3.19

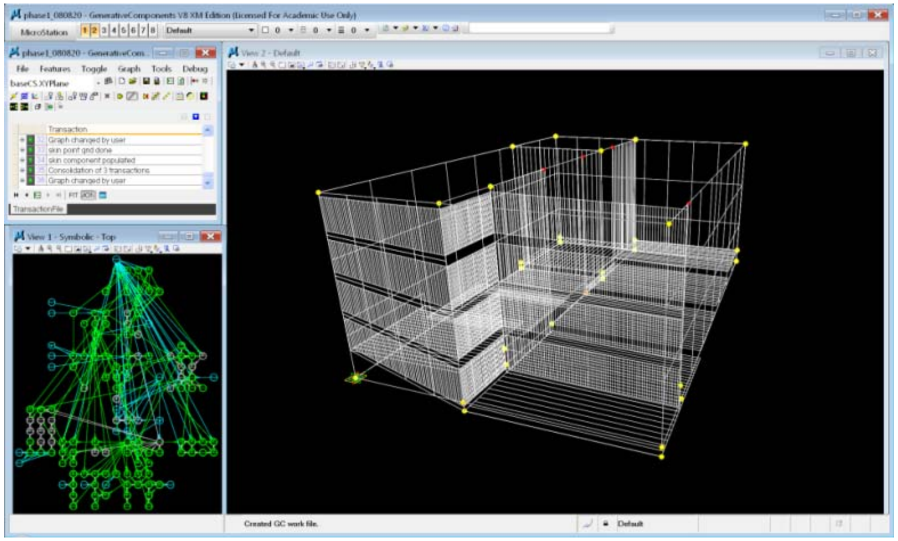
Populated skin component:

To generate skin variations, a model component is created as a surface that occupies each of the grid squares. Horizontal and vertical variables regulate the component's size. The populated component defines the shape and percentage of wall to glazing on the unit's outside surface.

Variable horizontal controls the horizontal length of surface component.

Variable vertical controls the vertical length of surface component.

```
transaction modelBased "terrace side component"
{
    feature point40 GC.Point
    {
        Surface                    = bsplineSurface08;
        U                          = Series(0,1,0.25);
        V                          = Series(0,1,0.25);
        SymbolXY                   = {94, 116};
        Replication                =
ReplicationOption.AllCombinations;
        Visible                    = false;
    }
    feature polygon03 GC.Polygon
    {
        Points                     = point40;
        SymbolXY                   = {94, 117};
        Visible                    = false;
    }
    feature terrace_depth GC.GraphVariable
    {
        Value                      = .1;
    }
    feature ubcskinComponent0803 GC.UBCSkinComponent08
    {
        horizontal                 = horizontal;
        polygon01                  = polygon03;
        vertical                   = vertical;
        SymbolXY                   = {94, 118};
    }
}
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
View 2 - Default
View 1 - Symbolic - Top
TransactionFile
Default

	Transaction
32	Graph changed by user
33	skin point grid done
34	skin component populated
35	Consolidation of 3 transactions
36	Graph changed by user

Created GC work file.

3.20

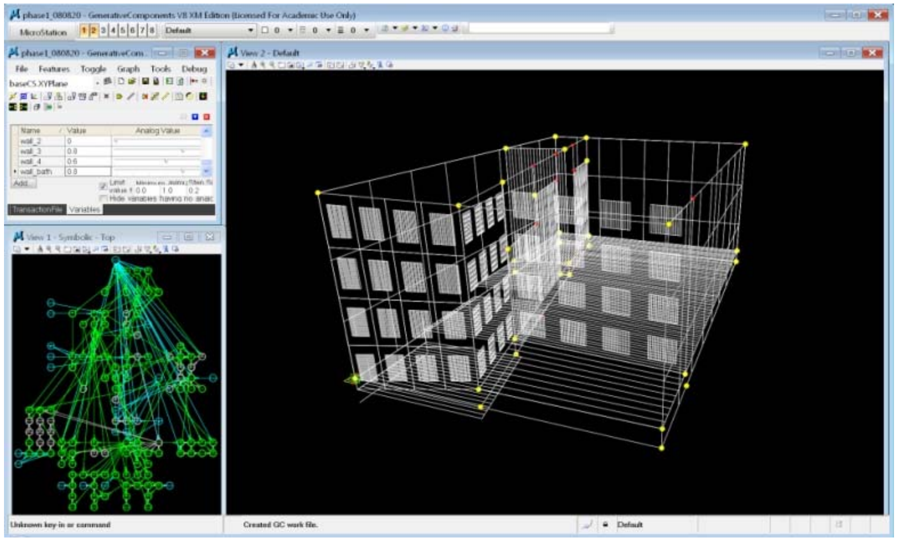
To allow for the deck space, the same component is populated on the deck's side, and the skin on the living space side is transferred to the wall separating it from the deck.

Variable horizontal controls the horizontal length of surface component.

Variable vertical controls the vertical length of surface component.

Their values are equal to the glazing length on the side of the skin component (measured as a percentage of the components vertical or horizontal size).

```
feature horizontal GC.GraphVariable
{
    Value                = .1;
}
feature vertical GC.GraphVariable
{
    Value                = 0;
```


phase1_080820 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
1 2 3 4 5 6 7 8
Default
phase1_080820 - GenerativeCom...
File Features Toggle Graph Tools Debug
baseCS.XYPlane
View 2 - Default
View 1 - Symbolic - Top
Add...
TransactionFile
Variables
Hide variables having no anal

Name	Value	Analog Value
wall_2	0	
wall_3	0.8	
wall_4	0.6	
wall_bath	0.8	

☑ Limit value 0.0 1.0 0.2

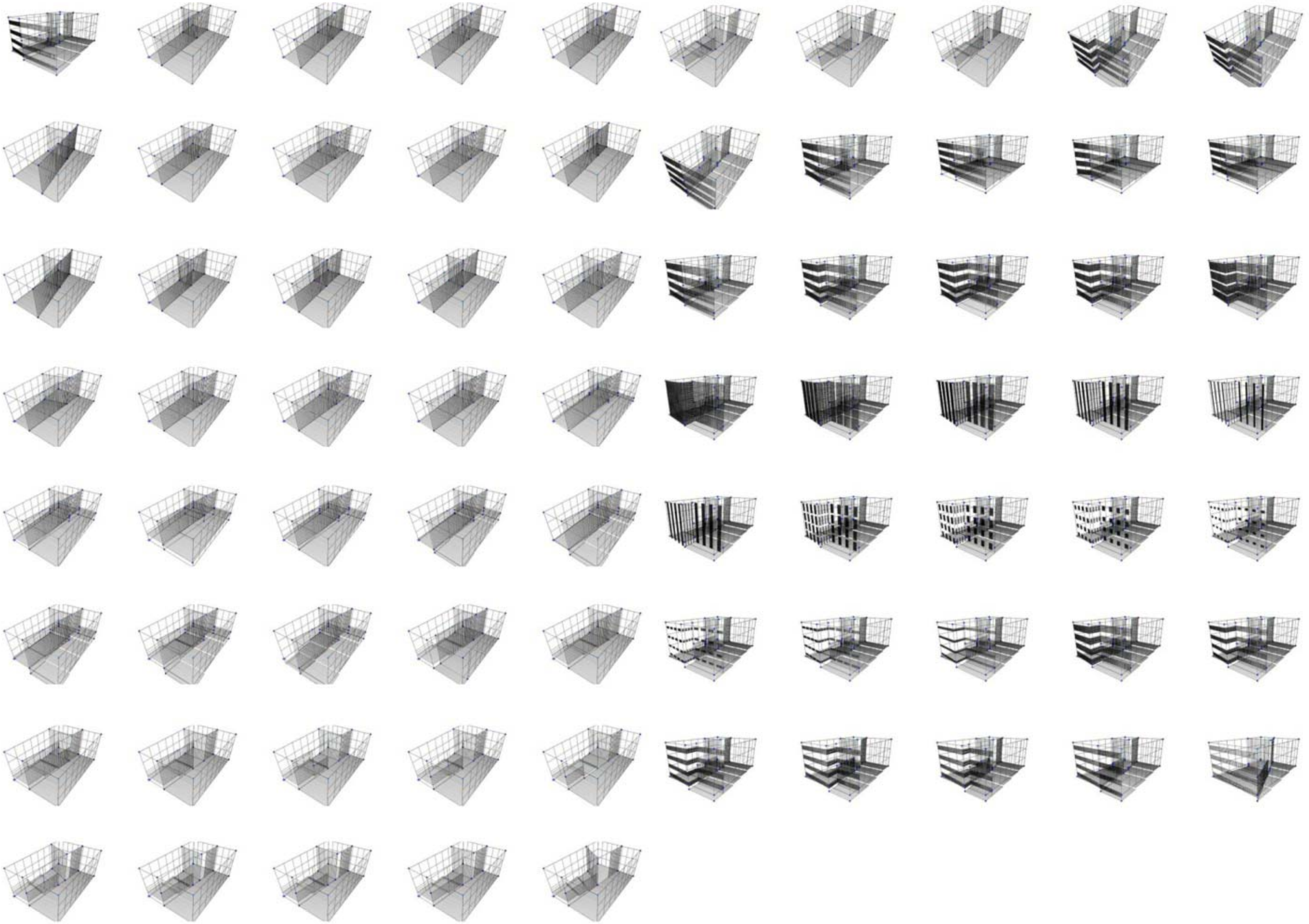
Unknown key-in or command

Created GC work file.

3.21

This screenshot shows the effect of manipulating vertical and horizontal lengths on each of the skin glazing components.

```
feature horizontal GC.GraphVariable
{
    Value                    = .2;
}
feature vertical GC.GraphVariable
{
    Value                    = 0.2;
}
```

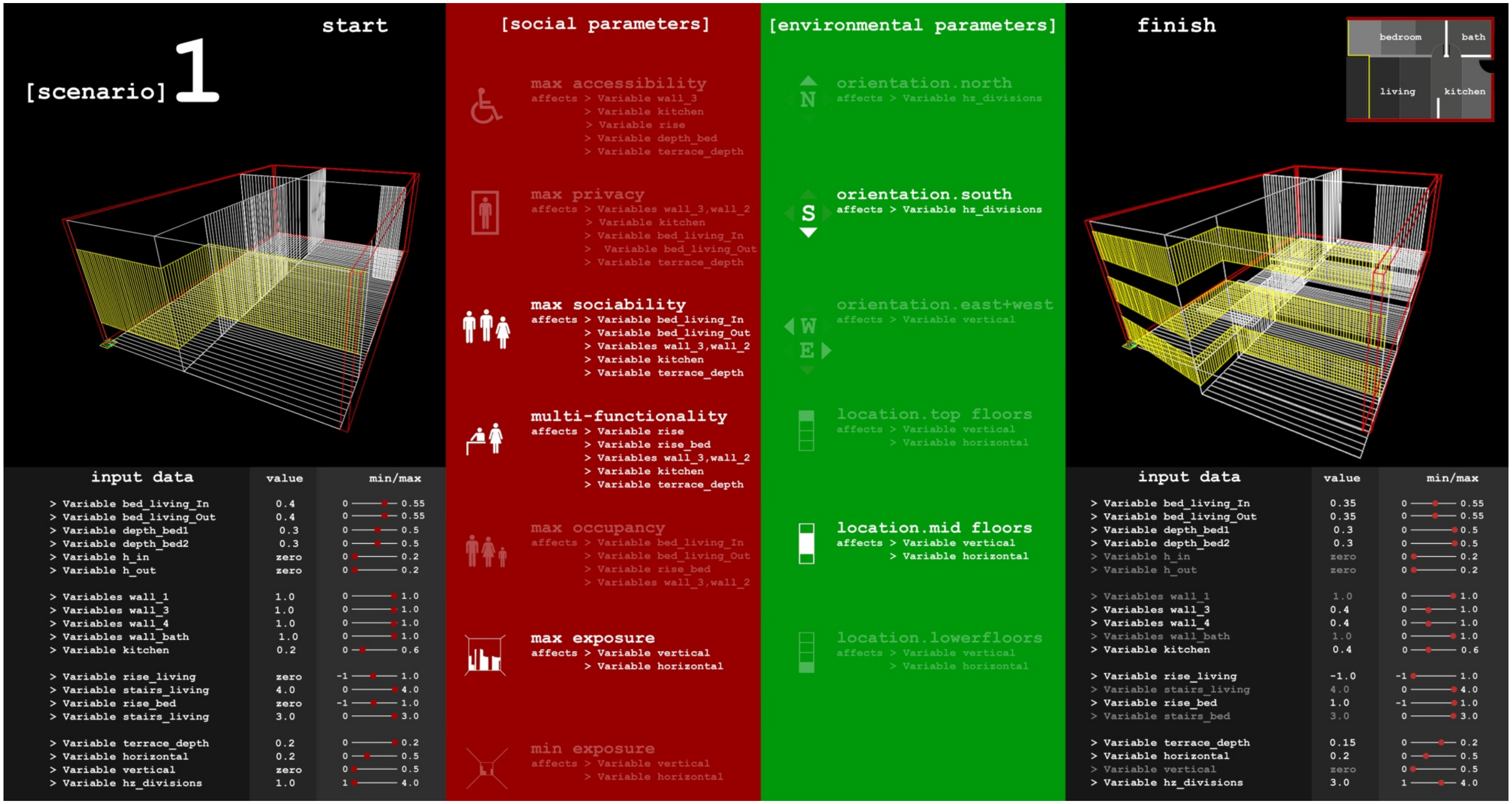


3.22 Instance of Model Configuration

Living Scenarios

I created a simple user interface based on the social and environmental factors, where potential users could choose the factors that best describe their needs. The system's variables are then manipulated to find the best configuration for the desired factors. Figures 3.24, 2.25, and 3.26 describe potential living scenarios inside the unit space by combining different social and environmental criteria. The illustrations show the values for the variables that reconfigure the system's spatial components according to the previously set rules. It is clear how the design system responds to the change in influences simultaneously and dynamically reconfigures its components to find a non-linear accumulation of spatial results.

For example, in scenario 1 (figure 3.24), a couple chooses the 'Maximum Sociability', 'Multi-Functionality', and 'Maximum Exposure' icons. Their unit is south facing and located in the middle floors of the tower. The input data are associated with these influencing factors and control the space configuration variables. 'Maximum Sociability' defines how the living space is bigger, the inside spaces are stepped to facilitate the 'Multi-Functionality', and the glazing is horizontally divided while maintaining a large area to accommodate for 'South Orientation' and 'Max Exposure'.



[scenario] 1

start

input data	value	min/max
> Variable bed_living_In	0.4	0 — 0.55
> Variable bed_living_Out	0.4	0 — 0.55
> Variable depth_bed1	0.3	0 — 0.5
> Variable depth_bed2	0.3	0 — 0.5
> Variable h_in	zero	0 — 0.2
> Variable h_out	zero	0 — 0.2
> Variables wall_1	1.0	0 — 1.0
> Variables wall_3	1.0	0 — 1.0
> Variables wall_4	1.0	0 — 1.0
> Variables wall_bath	1.0	0 — 1.0
> Variable kitchen	0.2	0 — 0.6
> Variable rise_living	zero	-1 — 1.0
> Variable stairs_living	4.0	0 — 4.0
> Variable rise_bed	zero	-1 — 1.0
> Variable stairs_living	3.0	0 — 3.0
> Variable terrace_depth	0.2	0 — 0.2
> Variable horizontal	0.2	0 — 0.5
> Variable vertical	zero	0 — 0.5
> Variable hz_divisions	1.0	1 — 4.0

[social parameters]

max accessibility
affects > Variable wall_3
> Variable kitchen
> Variable rise
> Variable depth_bed
> Variable terrace_depth

max privacy
affects > Variables wall_3,wall_2
> Variable kitchen
> Variable bed_living_In
> Variable bed_living_Out
> Variable terrace_depth

max sociability
affects > Variable bed_living_In
> Variable bed_living_Out
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

multi-functionality
affects > Variable rise
> Variable rise_bed
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

max occupancy
affects > Variable bed_living_In
> Variable bed_living_Out
> Variable rise_bed
> Variables wall_3,wall_2

max exposure
affects > Variable vertical
> Variable horizontal

min exposure
affects > Variable vertical
> Variable horizontal

[environmental parameters]

N orientation.north
affects > Variable hz_divisions

S orientation.south
affects > Variable hz_divisions

W E orientation.east+west
affects > Variable vertical

location.top floors
affects > Variable vertical
> Variable horizontal

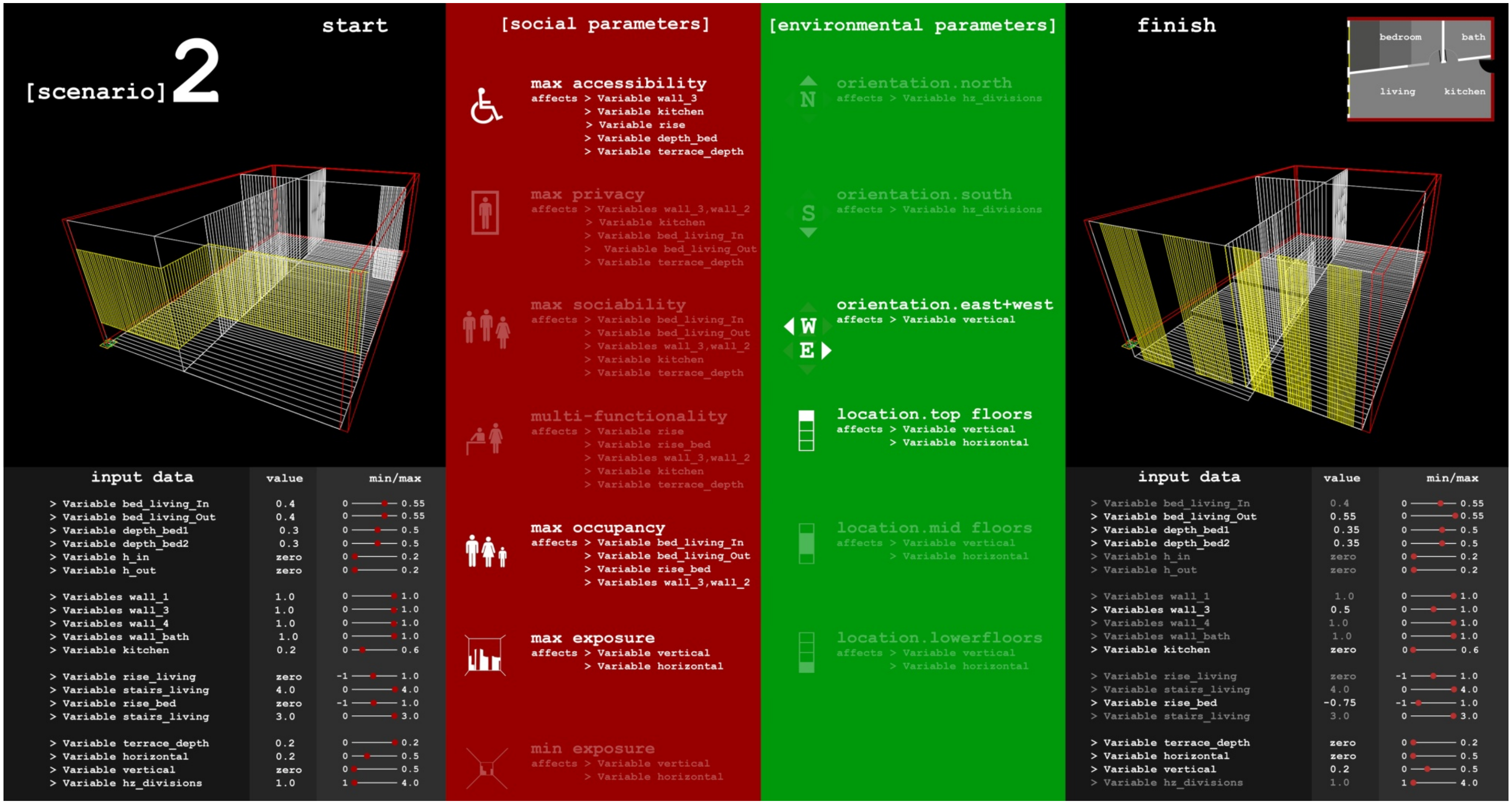
location.mid floors
affects > Variable vertical
> Variable horizontal

location.lowerfloors
affects > Variable vertical
> Variable horizontal

finish

input data	value	min/max
> Variable bed_living_In	0.35	0 — 0.55
> Variable bed_living_Out	0.35	0 — 0.55
> Variable depth_bed1	0.3	0 — 0.5
> Variable depth_bed2	0.3	0 — 0.5
> Variable h_in	zero	0 — 0.2
> Variable h_out	zero	0 — 0.2
> Variables wall_1	1.0	0 — 1.0
> Variables wall_3	0.4	0 — 1.0
> Variables wall_4	0.4	0 — 1.0
> Variables wall_bath	1.0	0 — 1.0
> Variable kitchen	0.4	0 — 0.6
> Variable rise_living	-1.0	-1 — 1.0
> Variable stairs_living	4.0	0 — 4.0
> Variable rise_bed	1.0	-1 — 1.0
> Variable stairs_bed	3.0	0 — 3.0
> Variable terrace_depth	0.15	0 — 0.2
> Variable horizontal	0.2	0 — 0.5
> Variable vertical	zero	0 — 0.5
> Variable hz_divisions	3.0	1 — 4.0

3.23 Living Scenario 1



[scenario] 2

start

input data	value	min/max
> Variable bed_living_In	0.4	0 – 0.55
> Variable bed_living_Out	0.4	0 – 0.55
> Variable depth_bed1	0.3	0 – 0.5
> Variable depth_bed2	0.3	0 – 0.5
> Variable h_in	zero	0 – 0.2
> Variable h_out	zero	0 – 0.2
> Variables wall_1	1.0	0 – 1.0
> Variables wall_3	1.0	0 – 1.0
> Variables wall_4	1.0	0 – 1.0
> Variables wall_bath	1.0	0 – 1.0
> Variable kitchen	0.2	0 – 0.6
> Variable rise_living	zero	-1 – 1.0
> Variable stairs_living	4.0	0 – 4.0
> Variable rise_bed	zero	-1 – 1.0
> Variable stairs_living	3.0	0 – 3.0
> Variable terrace_depth	0.2	0 – 0.2
> Variable horizontal	0.2	0 – 0.5
> Variable vertical	zero	0 – 0.5
> Variable hz_divisions	1.0	1 – 4.0

[social parameters]

max accessibility
affects > Variable wall_3
> Variable kitchen
> Variable rise
> Variable depth_bed
> Variable terrace_depth

max privacy
affects > Variables wall_3,wall_2
> Variable kitchen
> Variable bed_living_In
> Variable bed_living_Out
> Variable terrace_depth

max sociability
affects > Variable bed_living_In
> Variable bed_living_Out
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

multi-functionality
affects > Variable rise
> Variable rise_bed
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

max occupancy
affects > Variable bed_living_In
> Variable bed_living_Out
> Variable rise_bed
> Variables wall_3,wall_2

max exposure
affects > Variable vertical
> Variable horizontal

min exposure
affects > Variable vertical
> Variable horizontal

[environmental parameters]

N
orientation.north
affects > Variable hz_divisions

S
orientation.south
affects > Variable hz_divisions

W E
orientation.east+west
affects > Variable vertical

location.top floors
affects > Variable vertical
> Variable horizontal

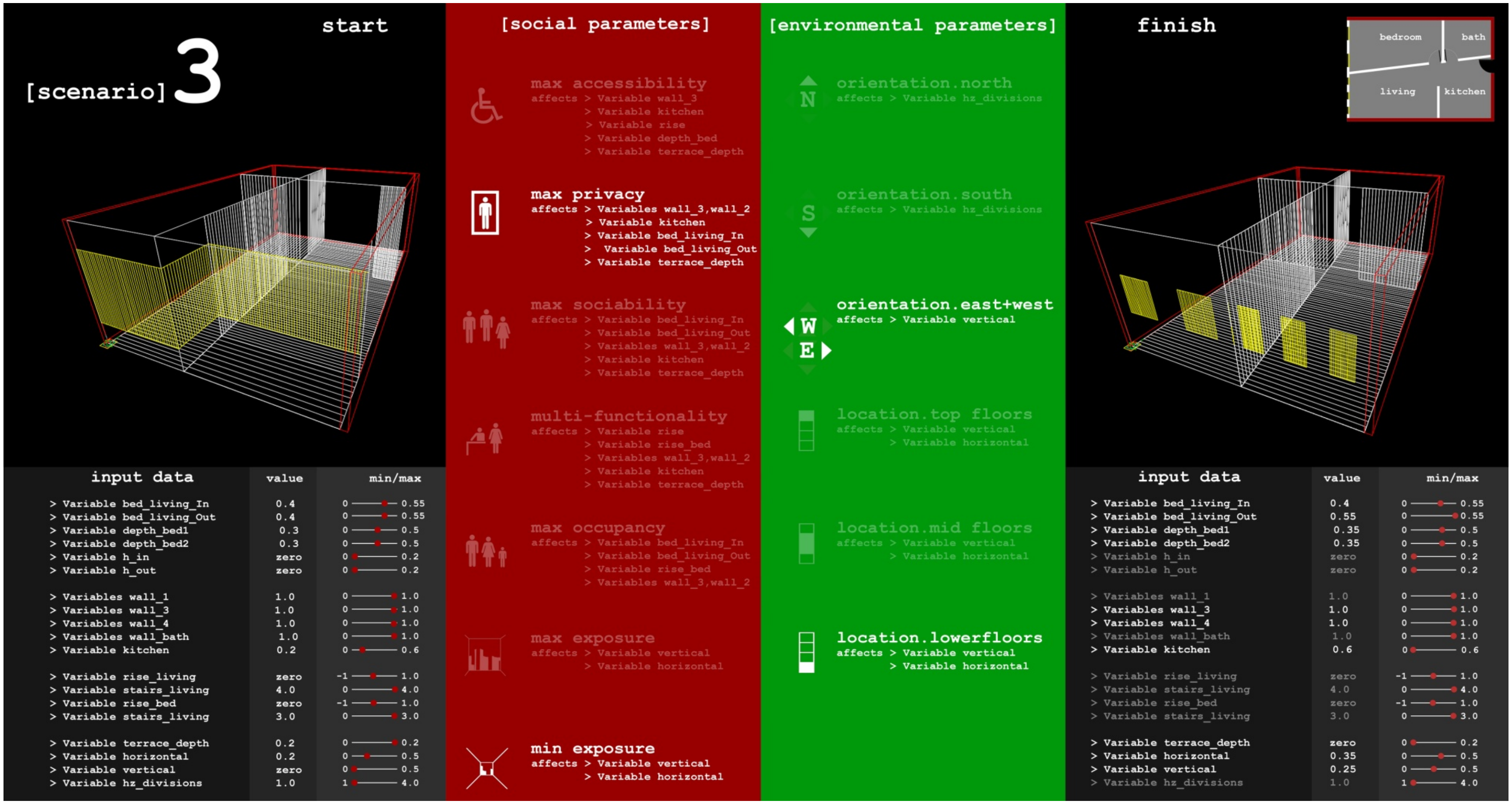
location.mid floors
affects > Variable vertical
> Variable horizontal

location.lowerfloors
affects > Variable vertical
> Variable horizontal

finish

input data	value	min/max
> Variable bed_living_In	0.4	0 – 0.55
> Variable bed_living_Out	0.55	0 – 0.55
> Variable depth_bed1	0.35	0 – 0.5
> Variable depth_bed2	0.35	0 – 0.5
> Variable h_in	zero	0 – 0.2
> Variable h_out	zero	0 – 0.2
> Variables wall_1	1.0	0 – 1.0
> Variables wall_3	0.5	0 – 1.0
> Variables wall_4	1.0	0 – 1.0
> Variables wall_bath	1.0	0 – 1.0
> Variable kitchen	zero	0 – 0.6
> Variable rise_living	zero	-1 – 1.0
> Variable stairs_living	4.0	0 – 4.0
> Variable rise_bed	-0.75	-1 – 1.0
> Variable stairs_living	3.0	0 – 3.0
> Variable terrace_depth	zero	0 – 0.2
> Variable horizontal	zero	0 – 0.5
> Variable vertical	0.2	0 – 0.5
> Variable hz_divisions	1.0	1 – 4.0

3.24 **Living Scenario 2**



start

[scenario] 3

input data	value	min/max
> Variable bed_living_In	0.4	0 — 0.55
> Variable bed_living_Out	0.4	0 — 0.55
> Variable depth_bed1	0.3	0 — 0.5
> Variable depth_bed2	0.3	0 — 0.5
> Variable h_in	zero	0 — 0.2
> Variable h_out	zero	0 — 0.2
> Variables wall_1	1.0	0 — 1.0
> Variables wall_3	1.0	0 — 1.0
> Variables wall_4	1.0	0 — 1.0
> Variables wall_bath	1.0	0 — 1.0
> Variable kitchen	0.2	0 — 0.6
> Variable rise_living	zero	-1 — 1.0
> Variable stairs_living	4.0	0 — 4.0
> Variable rise_bed	zero	-1 — 1.0
> Variable stairs_living	3.0	0 — 3.0
> Variable terrace_depth	0.2	0 — 0.2
> Variable horizontal	0.2	0 — 0.5
> Variable vertical	zero	0 — 0.5
> Variable hz_divisions	1.0	1 — 4.0

[social parameters]

max accessibility
affects > Variable wall_3
> Variable kitchen
> Variable rise
> Variable depth_bed
> Variable terrace_depth

max privacy
affects > Variables wall_3,wall_2
> Variable kitchen
> Variable bed_living_In
> Variable bed_living_Out
> Variable terrace_depth

max sociability
affects > Variable bed_living_In
> Variable bed_living_Out
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

multi-functionality
affects > Variable rise
> Variable rise_bed
> Variables wall_3,wall_2
> Variable kitchen
> Variable terrace_depth

max occupancy
affects > Variable bed_living_In
> Variable bed_living_Out
> Variable rise_bed
> Variables wall_3,wall_2

max exposure
affects > Variable vertical
> Variable horizontal

min exposure
affects > Variable vertical
> Variable horizontal

[environmental parameters]

N
orientation.north
affects > Variable hz_divisions

S
orientation.south
affects > Variable hz_divisions

W E
orientation.east+west
affects > Variable vertical

location.top floors
affects > Variable vertical
> Variable horizontal

location.mid floors
affects > Variable vertical
> Variable horizontal

location.lowerfloors
affects > Variable vertical
> Variable horizontal

finish

input data	value	min/max
> Variable bed_living_In	0.4	0 — 0.55
> Variable bed_living_Out	0.55	0 — 0.55
> Variable depth_bed1	0.35	0 — 0.5
> Variable depth_bed2	0.35	0 — 0.5
> Variable h_in	zero	0 — 0.2
> Variable h_out	zero	0 — 0.2
> Variables wall_1	1.0	0 — 1.0
> Variables wall_3	1.0	0 — 1.0
> Variables wall_4	1.0	0 — 1.0
> Variables wall_bath	1.0	0 — 1.0
> Variable kitchen	0.6	0 — 0.6
> Variable rise_living	zero	-1 — 1.0
> Variable stairs_living	4.0	0 — 4.0
> Variable rise_bed	zero	-1 — 1.0
> Variable stairs_living	3.0	0 — 3.0
> Variable terrace_depth	zero	0 — 0.2
> Variable horizontal	0.35	0 — 0.5
> Variable vertical	0.25	0 — 0.5
> Variable hz_divisions	1.0	1 — 4.0

3.25 Living Scenario 3

Comments:

The explorations in ‘Space Configurations’ mark a potential for the integration of procedural thinking into the way architecture is designed and conceived. The design of this exploring system offers a model that is responsive to various functional concerns in dense residential spaces. By using procedural thinking in the design, the components of space are attached to variables that can be manually controlled to initiate a global system reconfiguration. Various design decisions and constraints are achieved by the manipulation of each of the variables. It is also possible to design a digital user interface and directly link it to the system’s variables. In this case, potential space users can choose infinite combinations of occupation and location specific to their needs. The system automatically reconfigures itself using the information provided by users to generate the best working configuration for their living space.

In the light of this exploration, after-the fact evaluation and elimination of the non-working produced configurations is not needed for the designed responsive system. The rules that were set for the system for execution include the design limitations needed to control the system’s actions. This guarantees the production of working space configurations each time the system is manipulated.

In regards to the spatial possibilities offered by the system in this exploration, the system supports a wide array of configurations, which represent an accumulated progression of the designer’s preset possible system components. At the model scale, the system helps in the visualisation and manipulation of a range of designs within one model. This work model can function as a dynamic base for architects to start from to further explore the potentials that emerges from each instance of unit configurations.

The translation of design decisions into rules is greatly affected by my initial decision to address functional and programmatic concerns. Various limitations in spaces and their relationships are considered for different usability and accessibility reasons; which in turn reflect on the array of spatial configuration explored by the system.

As a result of exploring domestic spatial configurations inside the boundary of a living space-sized box, the designed system can be further utilised in the design of a residential high-rise; where user customized units affect the design of the whole structure on both the interior and exterior.

The digital techniques used in the design of this system are set to incorporate design constraints into the process of design. This approach is capable of producing responsive architecture that is capable of addressing a magnitude of design concerns, the only limits are what the designer feeds into the system. A rather simple design decision cumulatively turns into an increasingly complex one once the issues within the system multiply.

[7] City of Vancouver, Planning Department, Statistics and Information
http://vancouver.ca/commsvcs/planning/stats.htm
Accessed on: 28/10/2008, 2:38 PM

4 Tower Formation

Introduction

The scope of this exploration contributes to a design approach that makes use of computation and the logic of emergence to introduce form-finding techniques. In order to illustrate this approach to design, the exploration works on a building's scale in an urban setting. The goal is to generate form complexity out of a simple initial design, while dynamically responding at model scale to different forces that may influence a building's overall form. The design methodology relies on rule-based procedures throughout the entire exploration.

Design Approach

Instead of thinking of each design constraint or driver separately, and working through a regular design process to find the best solution for all; the model used in this exploration stores the design constraints and reacts to all of them simultaneously. This allows for a cumulative and complex effect of all the different influences that are applied to the model. With the right amount of abstraction, the designed model can address a great number of design references simultaneously. The designer has the control over how the model reacts to each reference by setting the rules for the model to follow. The most exciting aspect is that the model goes beyond representing how it is dynamically affected by each of the references, and actually calculates and represents how their effects interact. This computational modelling process reveals and solves additional layers of design complexity that are not possible by conventional CAD modeling.

Condominium housing in dense city cores is the context of this exploration. Parametric design techniques are applied to design a tower in downtown Vancouver, where the form is directly influenced by the tower's immediate context. A number of outside forces act on a designed dynamic system to generate the building's outer form. These outside forces correspond to a number of references selected to study their effects on the typical residential tower. Context, orientation and views are the references applied to the design model in this exploration.

It is important to mention that the proposed system is capable of addressing other issues relevant to high-rise residential design, which did not form a part of this study. As soon as the rules of how the system responds to each of the concerns are defined, the system calculates and graphically represents the influences, and cumulatively adds up the alteration to its form.

Design Background:

The exploration tackles the typical tower design in Vancouver's downtown core. While very successful in increasing density meeting the city's planning goals, and satisfying developers, residential tower design has not contributed much to architectural design and innovation in the city. It has produced much of repetition in the way architectural form is designed and perceived.

The design approach proposed in this thesis provides a conceptual alternative to an existing high-rise building. It uses the environment in context to drive the design process, and perform an accountable role in defining the tower's form.

Objective:

The objective is to create a dynamic responsive system able to interact with the building’s immediate environment at model scale. The system should successfully use these dynamic interactions to control the building's form, and adjust its dependent components according to changes in the influencing factors.

The system success is tested through its adaptation abilities; it should ultimately be responsive within the dynamic relationships with its outside urban context and its use, and should solve increasingly complex relationships to generate the building's form.

Site:

The proposed site is located at Yaletown Park at the intersection of Nelson and Mainland streets in Downtown Vancouver, BC. The existing site hosts three recently constructed towers and a public park.

The site provides a rich context for exploration, and is chosen for the following reasons:

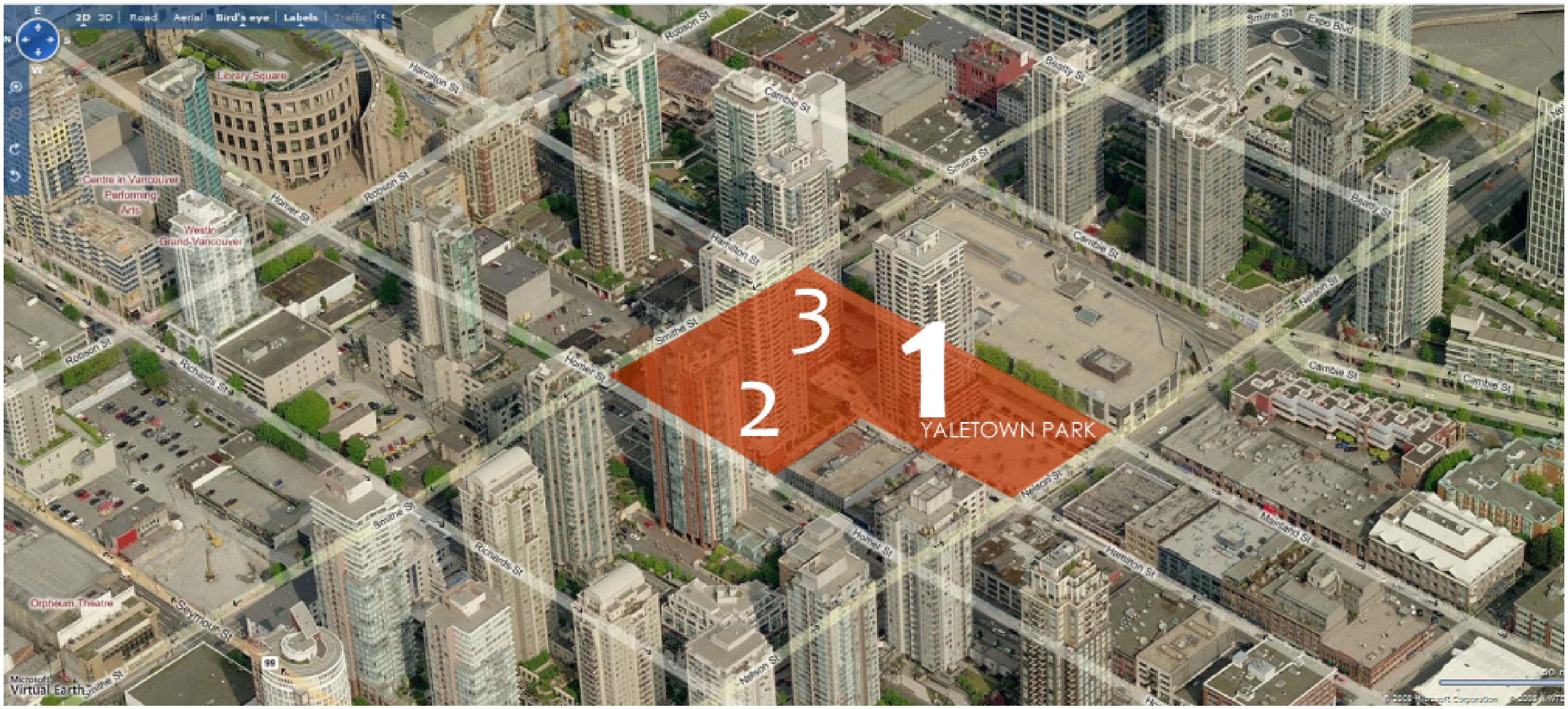
1. Its location inside the downtown core with various high-rise and low-rise buildings in its immediate context.

2. The site offers a number of views to significant parts of the city, which are used to investigate how views can affect the building's form.

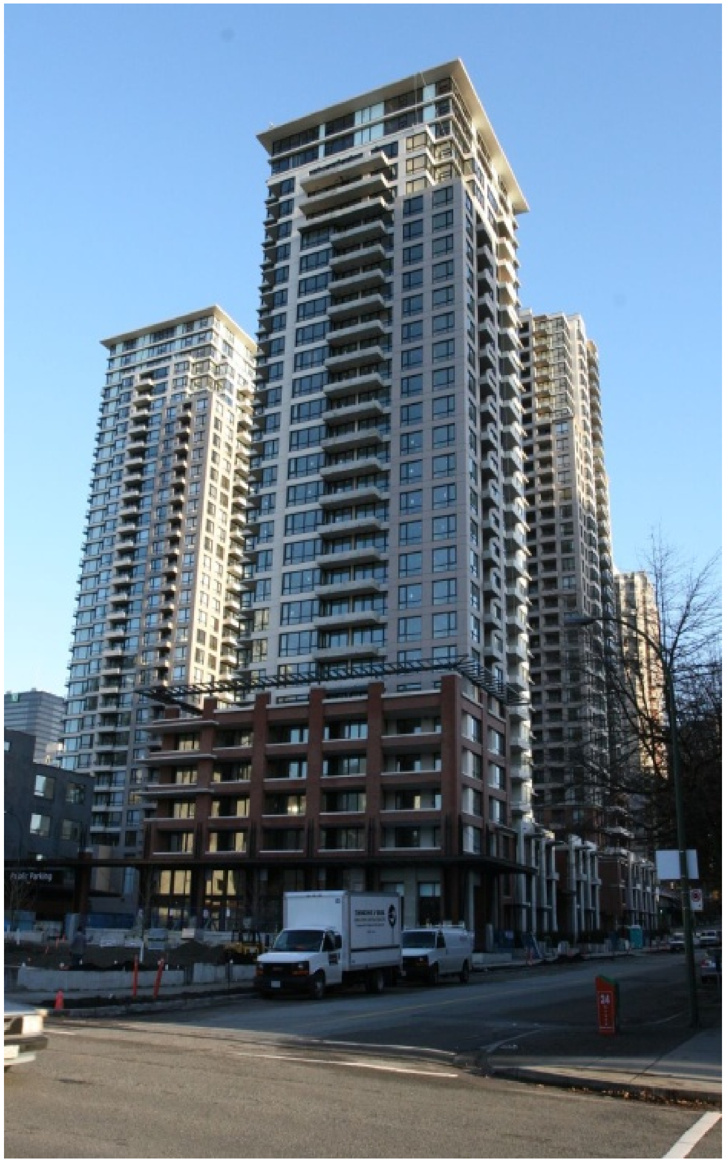
3. The site is crossed by View Corridor 9.2.1 which is assigned by the City of Vancouver as a protected view. This provides a potential to inspect the relationship between a building and a crossing view corridor.

4. The site includes a park within its boundary. The relationship between a park and its immediate neighbouring building is explored in this model.

5. The site is composed of three different towers, this provides a potential to study the relationships between them in further explorations.



4.1 Site



4.2 Existing Architecture on Site

Existing Architecture: Designed by Buttjes Architecture, it qualifies to be studied as typical of downtown Vancouver. The typical building consists of a 30 storey tower mostly containing small and efficient modern one bedroom suites and a townhouses podium to provide a more intimate feeling at the street level. It is successful in meeting the city's and the market requirements with its amenities, finishes, and efficient use of areas and spaces.

There seem to be very little account for location and orientation. The living units are designed for maximum exposure regardless of their location within the building. The building mass can be easily reduced to an extruded square, which implies minimal context response.

Position:

A dynamic form-finding system is designed in this exploration. It uses the existing program, site and context, and examines it by taking the following factors into account at both form and skin scales:

1. View corridor

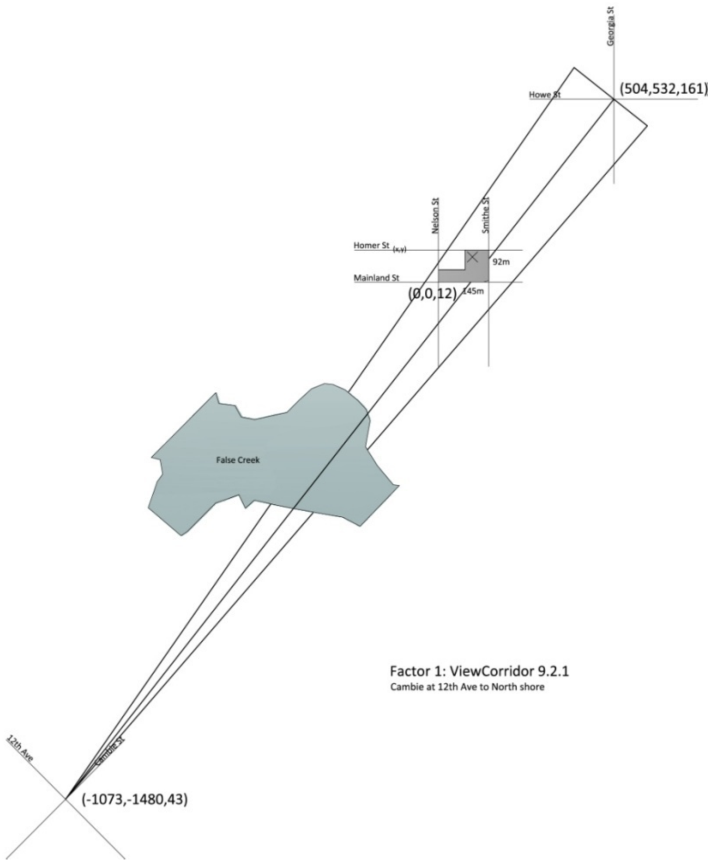
2. Orientation

3. Context

Methodology:

Parametric modelling in *GC* is used as a platform to tackle the complexity of this design and to set and follow the rules through which a form is generated. The design problem dramatically increases in complexity as the factors taken into account change or increase in a dynamic dialogue between the influencing factors and the tower's form.

The proposed form starts with a high degree of density achieved by maintaining the original building's volume, while offering substantial spatial and programmatic change attained by adjusting to the surrounding contextual forces. These forces cause deformation in the building's skin, which in turn results in changing the inside space organisation and plan layouts. This approach ultimately leads to a building form that responds intelligently to surroundings. The exploration being at model scale provides a huge playground for generating a building form that follows the designed rules. The resulting form should also result in an equally complex mixture of spaces, room sizes and apartment organisations, which should serve our main goal of introducing new possibilities for occupation in the city. However, the interior reflections of this tower design are not explored in this thesis.



4.3 View Corridor 9.2.1

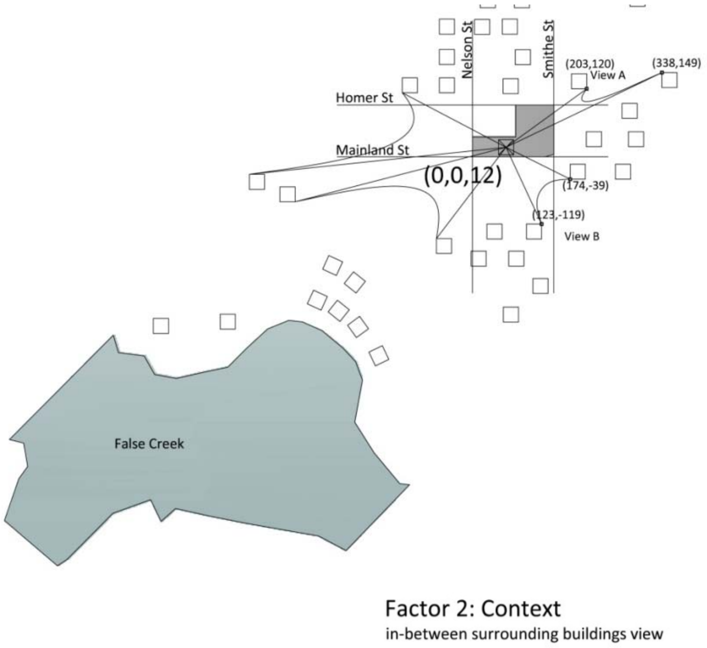
Design Process:

The high-rise tower is designed as a dynamic model in *GenerativeComponents*. The initial tower is determined by respecting the directions of adjacent streets, applying setbacks, and extruding a building volume. The dynamic design system is attached to different pressures and forces that come from the surrounding structures, sun exposure, and crossing view corridors. The system is set to respond to changes within these influencing forces in real time, thus allowing for experimentation at model scale.

For each of these factors, rules are designed and set inside the software to regulate the building's reaction to each of the forces.

Initial Form: The initial form of the tower is the most basic extruded rectangle that comes out from considering the site streets, setbacks, and the allowable building height.

Factor 1: View Corridors: The City of Vancouver assigned a number of view corridors in 1991 to preserve certain views to the North Shore Mountains from a number of different locations across the city[8]. Building heights and forms are required by the City to provide room for the view cones to stretch without obstruction. The proposed design system reacts to view cones crossing its site, not only by adjusting heights and forms, but also by engaging the building with the preserved mountains view. When a view cones crosses a site, the building reacts to avoid obstruction, and changes its form to offer as much exposure to the direction of the view as possible, while also maintaining a relatively constant building volume.



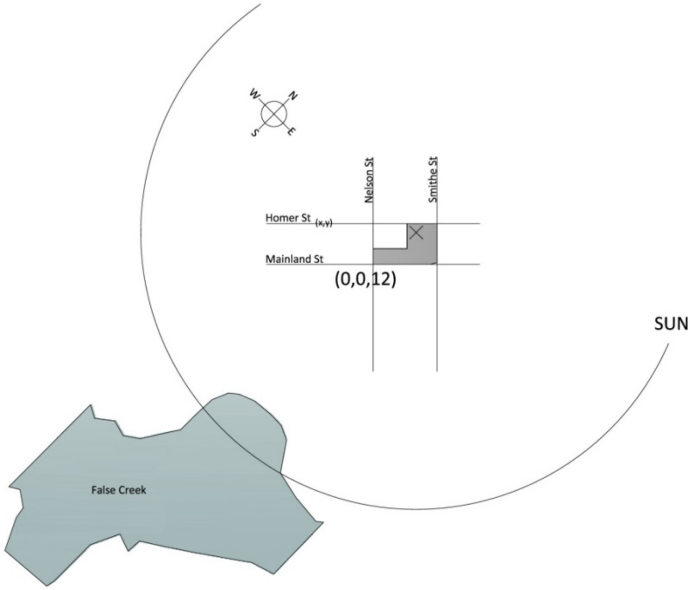
4.4 Context at Form Scale 1

4.5 Context at From Scale 2

Factor 2: Context at Form Scale:

The site is surrounded by a number of high-rise and low-rise buildings. In addition to the preserved view corridor, the spaces between the surrounding buildings offer open views to different parts of the city. Two prospective in-between views were identified in relation to the tower, and were used to force the tower's form.

In order to engage with these views, a V shaped cut is done on the tower's faces to provide more surface exposure towards these particular in-between spaces. The width of the cuts is proportional to the dimension of the in-between spaces. The V cut is deeper at the tower's top providing more exposure at the higher levels.



Factor 3: Orientation

4.6 Orientation

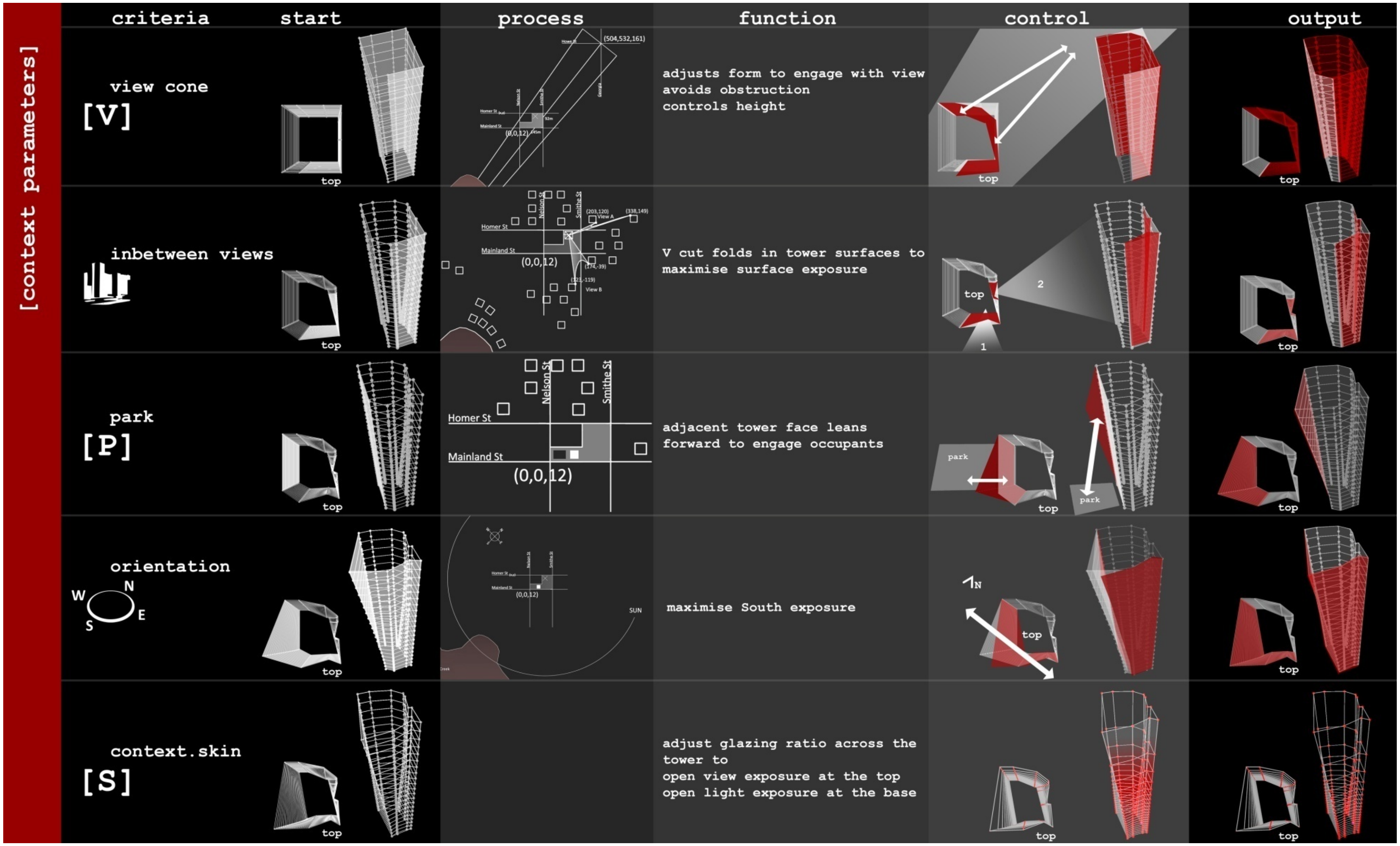
Factor 3: Orientation:

The system is designed to maximise south exposure, making use of sun rotation and synchronizing the form to its orientation. Efficiency is considered in seeking the maximum passive solar heating, and the building's form will adjust to maximise its south-exposed surface while maintain a relatively constant building volume.

Factor 4: Context at Skin Scale:

The system should work to allow open view exposure at the top of the building, and more reserved ones where the building is immediately adjacent to another one. At the same time, floors near the base should have open light exposures to accommodate for decreasing lighting quality towards the base. Facade components are used on the surface skin, where the facade units dynamically accommodate the changes in spacing required by its references.

The above-mentioned factors contribute collectively to a complex process of smart form finding. They are tested in a dynamic system of references, and each of the factors has different levels of influence on the building's form. The system will respond to each of these factors at the same time, and determine how they affect each other and the building's form. This design approach is more than an additive process of forces; it gains its power from collectively considering a number of components in a dynamic environment.



adjusts form to engage with view
avoids obstruction
controls height

V cut folds in tower surfaces to maximise surface exposure

adjacent tower face leans forward to engage occupants

maximise South exposure

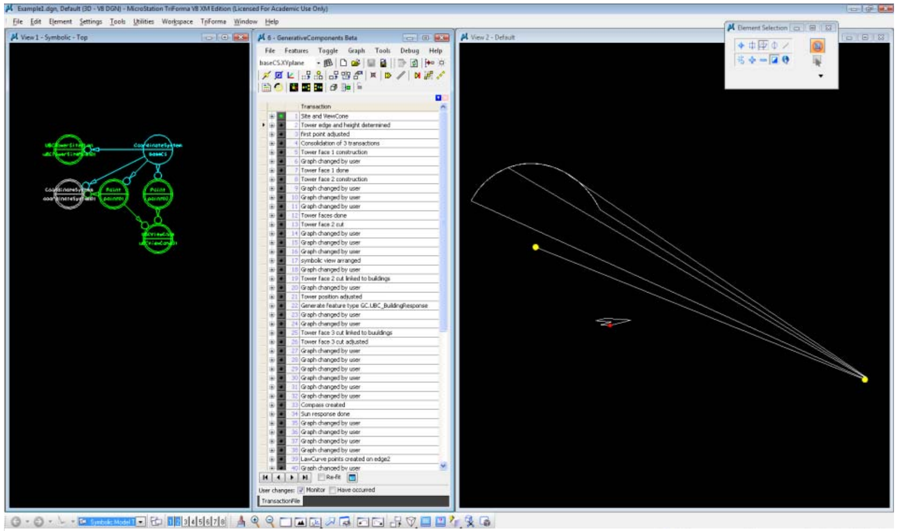
adjust glazing ratio across the tower to
open view exposure at the top
open light exposure at the base

4.7 Tower Formation Process

Figure 4.7 describes the process of geometrical formation of the tower. The diagrams show how the tower's form responds to each of the individual influencing criteria (factors).

System Making

The following describes the process of creating the parametric design system in *GenerativeComponents*. The Model unit's components are modeled together with the rules, constraints, and variables that control the interaction between components.


Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
Element Selection
Symbolic Model 1

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Tower face 1 done
8	Tower face 2 construction
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Tower faces done
13	Tower face 2 cut
14	Graph changed by user
15	Graph changed by user
16	Graph changed by user
17	symbolic view arranged
18	Graph changed by user
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generate feature type GC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buuildings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user

□ Re-fit

User changes: ☑ Monitor □ Have occurred

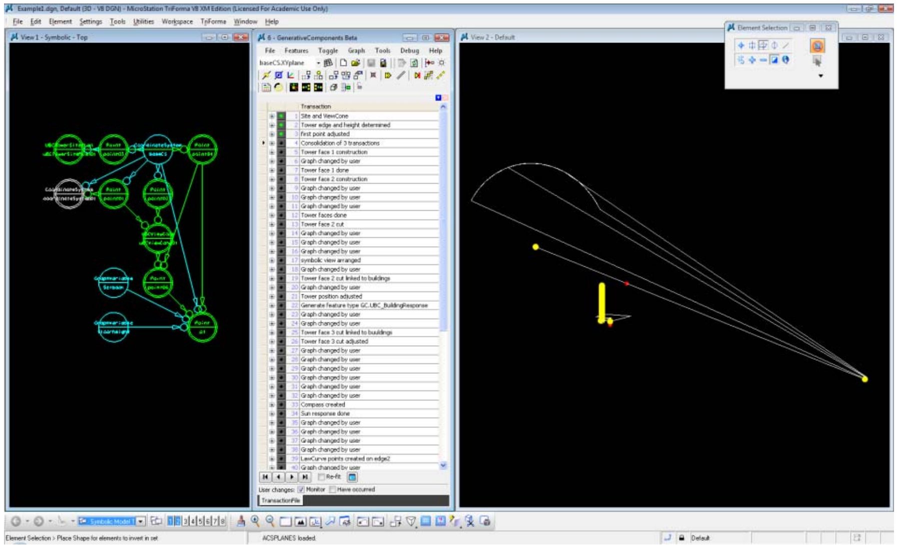
TransactionFile

4.8

Site Specifications:
Site starting point: Nelson at Mainland St
Site dimensions: 145 m x 92 m
Site elevation: 12 m average
View corridor 9.2.1 is drawn in relation to the site location.

View Cone: Reference data:
Preserved view: North Shore Mountains
View Point: Cambie at 12th Avenue (43.47 m elevation)
Reference Point 9.2.1: T.D. Bank Tower (161.73 m elevation)
Distance from view point to reference point: 2419.80 m

```
transaction modelBased "Tower edge and height determined"
{
    feature Setback GC.GraphVariable
    {
        Value                       = 3.0;
        LimitValueToRange           = true;
        RangeMaximum                = 10.0;
        RangeStepSize               = 1.0;
        SymbolXY                    = {99, 103};
    }
    feature floorheight GC.GraphVariable
    {
        Value                       = 3;
        LimitValueToRange           = true;
        RangeMinimum                = 2.5;
        RangeMaximum                = 5.0;
        RangeStepSize               = 0.5;
        SymbolXY                    = {99, 104};
    }
    feature p1 GC.Point
    {
        CoordinateSystem            = baseCS;
        Xtranslation                = point04.X+Setback;
        Ytranslation                = point04.Y-Setback;
        Ztranslation                =
Series(point04.Z,point06.Z,floorheight);
        SymbolXY                    = {101, 104};
        HandleDisplay               = DisplayOption.Display;
    }
    feature point03 GC.Point
    {
        CoordinateSystem            = baseCS;
        Xtranslation                = 0;
        Ytranslation                = 0;
        Ztranslation                = 12;
        SymbolXY                    = {99, 100};
        HandleDisplay               = DisplayOption.Display;
    }
    feature point04 GC.Point
    {
        CoordinateSystem            = baseCS;
        Xtranslation                =
uBCTowerSitePLan01.line04.StartPoint.X;
        Ytranslation                =
uBCTowerSitePLan01.line04.StartPoint.Y;
        Ztranslation                =
uBCTowerSitePLan01.line04.StartPoint.Z;
        SymbolXY                    = {101, 100};
        HandleDisplay               = DisplayOption.Display;
    }
    feature point06 GC.Point
    {
        Curve                       = uBCViewCone01.View;
        PointToProjectOntoCurve     = point04[0];
        SymbolXY                    = {100, 103};
    }
    feature uBCTowerSitePLan01 GC.UBCTowerSitePLan
    {
        SiteOrigin                  = point03;
    }
}

transaction modelBased "first point adjusted"
{
    feature point04 GC.Point
    {
        Xtranslation                =
uBCTowerSitePLan01.line06.StartPoint.X;
        Ytranslation                =
uBCTowerSitePLan01.line06.StartPoint.Y;
        Ztranslation                =
uBCTowerSitePLan01.line06.StartPoint.Z;
    }
}

transaction modelBased "Consolidation of 3 transactions"
{
    feature Parkdepth GC.GraphVariable
    {
        Value                       = 35;
        LimitValueToRange           = true;
        RangeMaximum                = 40.0;
        RangeStepSize               = 0.0;
    }
    feature p1 GC.Point
    {
        Xtranslation                = point04.X+Setback+Parkdepth;
    }
```


Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)

6 - GenerativeComponents Beta

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Tower face 1 done
8	Tower face 2 construction
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Tower faces done
13	Tower face 2 cut
14	Graph changed by user
15	Graph changed by user
16	Graph changed by user
17	symbolic view arranged
18	Graph changed by user
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generate feature type GC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buuldings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user

User changes: ☑ Monitor ☐ Have occurred

Element Selection > Place Shape for elements to invert in set

ACSPLANES loaded

View 1 - Symbolic - Top
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
Re-fit
TransactionFile
View 2 - Default
Element Selection
Default

4.9

First Edge:

The tower's first edge p1 is created. It is placed within a variable distance setback from the site's corner.

Height: is determined by the difference in elevation between:

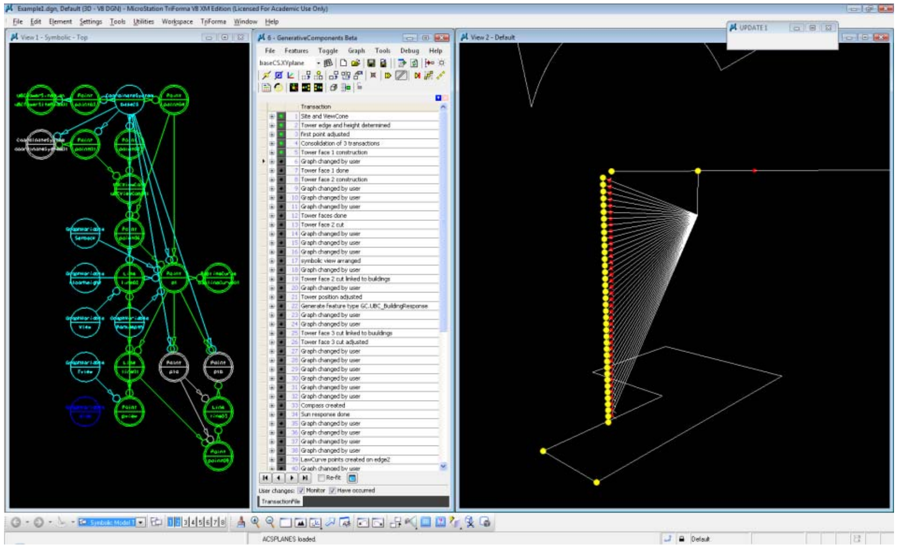
a. The Site's projection on the view corridor, and

b. The site's plane at 12 m

Floor heights: is set as a global variable.

When the floor height is 3 meters, the calculated number of floors equals 40.

```
transaction modelBased "Tower face 1 construction"
{
    feature Setback GC.GraphVariable
    {
        RangeMaximum              = 1.0;
    }
    feature Tview GC.GraphVariable
    {
        Value                     = 0.76;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {99, 106};
    }
    feature View GC.GraphVariable
    {
        Value                     = 26.0;
        LimitValueToRange         = true;
        RangeStepSize             = 0.0;
        SymbolXY                  = {99, 105};
    }
    feature bsplineCurve01 GC.BsplineCurve
    {
        FitPoints                 = {p1};
        SymbolXY                  = {102, 104};
    }
    feature line01 GC.Line
    {
        StartPoint                = p1;
        EndPoint                  = line02.EndPoint;
        SymbolXY                  = {100, 106};
    }
    feature line02 GC.Line
    {
        StartPoint                = pview;
        Direction                 = baseCS.Zdirection;
        Length                    = -View;
        SymbolXY                  = {100, 104};
    }
    feature line03 GC.Line
    {
        StartPoint                = point09;
        EndPoint                  = p1a;
        SymbolXY                  = {102, 107};
    }
    feature p1a GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = p1.X+5;
        Ytranslation              = p1.Y;
        Ztranslation              = p1.Z;
        SymbolXY                  = {101, 106};
        HandleDisplay             = DisplayOption.Display;
    }
    feature p1b GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = p1.X+20;
        Ytranslation              = p1.Y-line03.Length;
        Ztranslation              = p1.Z;
        SymbolXY                  = {102, 106};
    }
    feature point09 GC.Point
    {
        Curve                     = line01;
        PointToProjectOntoCurve   = p1a;
        SymbolXY                  = {102, 108};
    }
    feature pview GC.Point
    {
        Curve                     = uBCViewCone01.View;
        T                         = Tview;
        SymbolXY                  = {100, 107};
        HandleDisplay             = DisplayOption.Display;
    }
    feature side GC.GraphVariable
    {
        Value                     = 18;
        LimitValueToRange         = true;
        RangeMinimum              = 10.0;
        RangeMaximum              = 30.0;
        RangeStepSize             = 1.0;
        SymbolXY                  = {99, 107};
    }
}
```

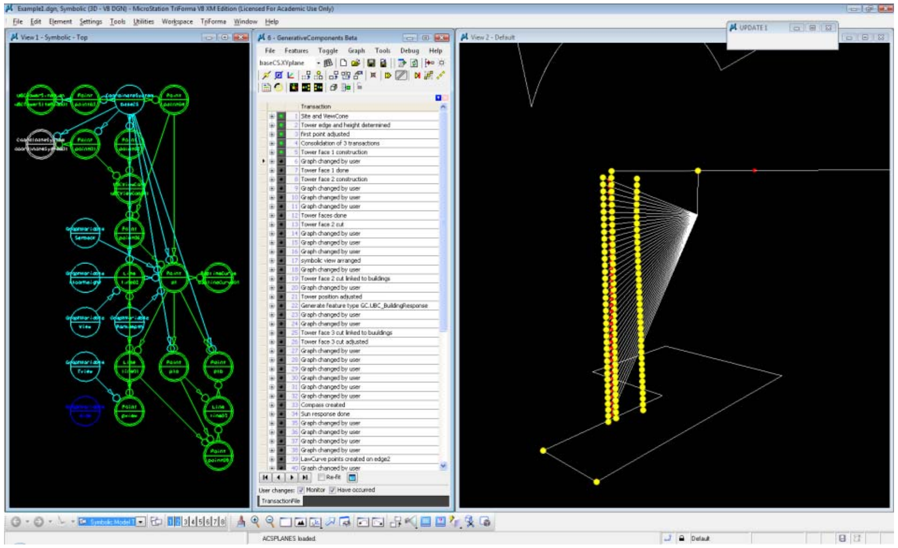

Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
UPDATE 1
Re-fit
User changes: Monitor Have occurred
TransactionFile
ACSPLANES loaded
Default

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Tower face 1 done
8	Tower face 2 construction
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Tower faces done
13	Tower face 2 cut
14	Graph changed by user
15	Graph changed by user
16	Graph changed by user
17	symbolic view arranged
18	Graph changed by user
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generate feature type GC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buildings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user

4.10

The tower's edge point series p1 is connected by a line line01 to the view corridor's reference point (target).

Tview and View are variables connected to the target point of the View Cone. Their manipulation fine tunes the tower's orientation towards the view target.



Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licenced For Academic Use Only)
View 1 - Symbolic - Top
8 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
UPDATE 1
Re-fit
User changes: Monitor Have occurred
TransactionFile
ACSPLANES loaded
Default

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Tower face 1 done
8	Tower face 2 construction
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Tower faces done
13	Tower face 2 cut
14	Graph changed by user
15	Graph changed by user
16	Graph changed by user
17	symbolic view arranged
18	Graph changed by user
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generate feature type GC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buuildings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user

4.11

View cone response rule:

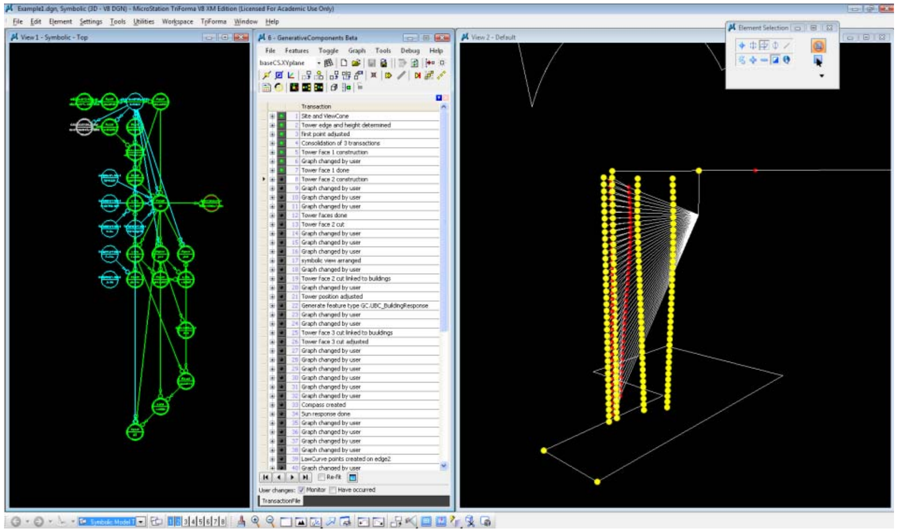
The tower's faces should adjust to better engage with the view. The points defining each of the tower's faces should move to provide greater face exposure to the view target.

Each face is defined by 4 series of points:

2 Edges + 2 Isocurves on the face

The second Isocurve's p1b position moves backward (Y-direction) to direct the face towards the view target. The distance moved equals the distance between the first Isocurve p1a and line01. (Figure 4.11)

```
transaction modelBased "Graph changed by user"
{
    feature View GC.GraphVariable
    {
        Value                     = 29;
    }
}

transaction modelBased "Graph changed by user"
{
    feature line04 GC.Line
    {
        StartPoint                = point11;
        EndPoint                  = p1a;
    }
    feature line04_EndPoint GC.Point
    {
        Plane                     = baseCS.XYplane;
        Xtranslation              = <free> (507.950808438218);
        Ytranslation              = <free> (460.042810956901);
    }
    feature p2 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = p1.X+30;
        Ytranslation              = p1.Y-line04.Length;
        Ztranslation              = p1.Z;
        HandleDisplay             = DisplayOption.Display;
    }
    feature point11 GC.Point
    {
        Curve                     = line01;
        PointToProjectOntoCurve   = p1b;
    }
}

transaction modelBased "Tower face 1 done"
{
    deleteFeature line04_EndPoint;
    feature Tview GC.GraphVariable
    {
        Value                     = 0.7599;
        RangeMinimum              = 0.75;
        RangeMaximum              = 0.78;
    }
    feature bsplineCurve01 GC.BsplineCurve
    {
        SymbolXY                  = {103, 104};
    }
    feature c1a GC.BsplineCurve
    {
        FitPoints                 = {p1a};
        SymbolXY                  = {102, 109};
    }
    feature point09 GC.Point
    {
        SymbolXY                  = {101, 107};
    }
}
```


Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
Element Selection

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower Face 1 construction
6	Graph changed by user
7	Tower Face 1 done
8	Tower Face 2 construction
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Tower faces done
13	Tower Face 2 cut
14	Graph changed by user
15	Graph changed by user
16	Graph changed by user
17	symbolic view arranged
18	Graph changed by user
19	Tower Face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generate feature type GC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower Face 3 cut linked to buildings
26	Tower Face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user

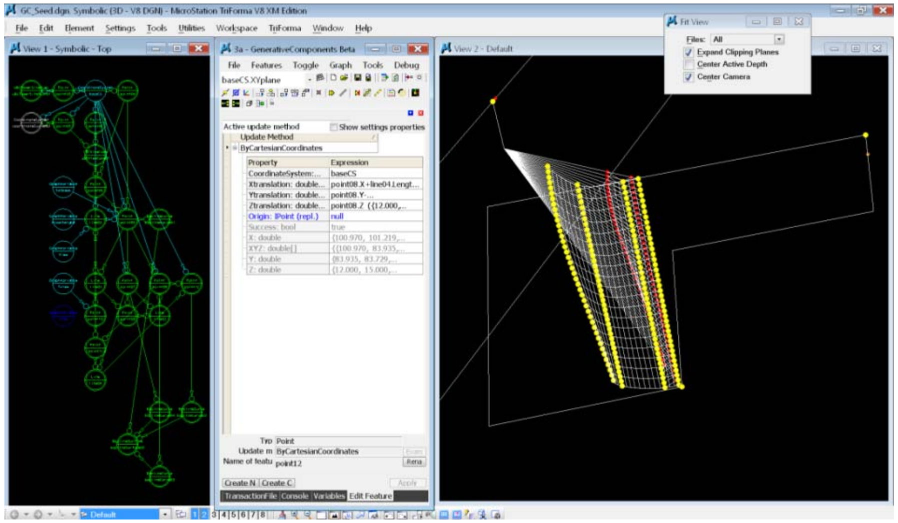
□ Re-fit

User changes: ☑ Monitor □ Have occurred

TransactionFile

4.12

A similar move happens at the second edge of the tower p2, where it moves backwards (Y-direction). The distance moved equals the distance between the second Isocurve p1b and line01. Side is a variable that determines the building's face width. Spacing between each of the face curves is a factor of this variable.


GC_Seed.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
3a - GenerativeComponents Beta
File Features Toggle Graph Tools Debug
baseCS.XYplane
View 2 - Default
Fit View
Default

Active update method □ Show settings properties

Update Method: ByCartesianCoordinates

Property	Expression
CoordinateSystem:...	baseCS
Xtranslation: double...	point08.X+line04.Lengt...
Ytranslation: double...	point08.Y-...
Ztranslation: double...	point08.Z ({12.000,...
Origin: IPoint (repl.)	null
Success: bool	true
X: double	{100.970, 101.219,...
XYZ: double[]	{{100.970, 83.935,...
Y: double	{83.935, 83.729,...
Z: double	{12.000, 15.000,...

Typ: Point

Update m: ByCartesianCoordinates

Name of featu: point12

Create N | Create C | Apply

TransactionFile | Console | Variables | Edit Feature

Files: All

☑ Expand Clipping Planes

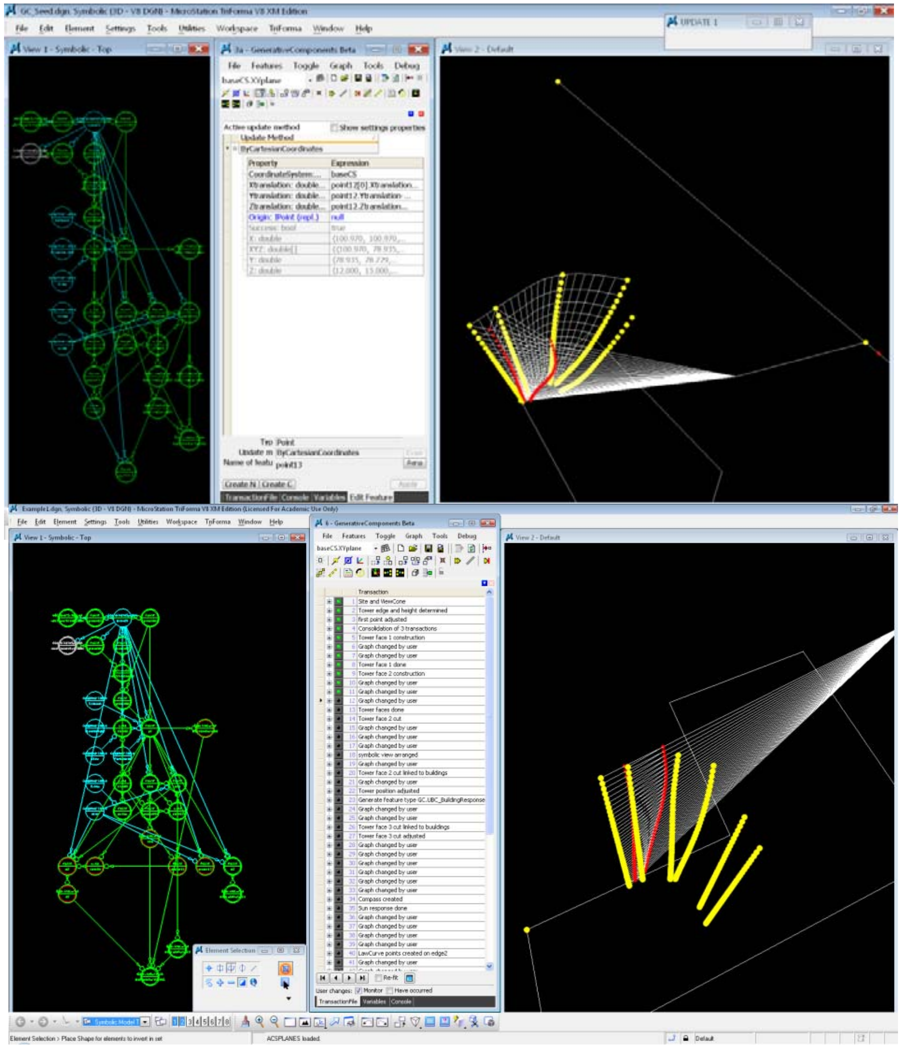
□ Center Active Depth

☑ Center Camera

4.13

A surface mesh s12 lofts the curves defining the first face of the tower.

```
transaction modelBased "Tower face 2 construction"
{
    feature p1a GC.Point
    {
        Xtranslation              = p1.X+side*.25;
    }
    feature p1b GC.Point
    {
        Xtranslation              = p1a.X+side*.45;
    }
    feature p2 GC.Point
    {
        Xtranslation              = p1b.X+line04.Length*2-side*.6;
        Ytranslation              = p1b.Y-line04.Length*2+side*.65;
    }
    feature p3 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = p1.X+side;
        Ytranslation              = p1.Y-side;
        Ztranslation              = p1.Ztranslation;
    }
    feature point13 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = p1.X+side;
        Ytranslation              = p1.Y-side*.75;
        Ztranslation              = p1.Ztranslation;
    }
    feature side GC.GraphVariable
    {
        Value                     = 20;
    }
}

transaction modelBased "Graph changed by user"
{
    feature bsplineCurve12 GC.BsplineCurve
    {
        FitPoints                 = {p1b};
    }
    feature bsplineCurve14 GC.BsplineCurve
    {
        FitPoints                 = {p3};
        SymbolXY                  = {104, 110};
    }
    feature c1a GC.BsplineCurve
    {
        SymbolXY                  = {101, 108};
    }
    feature c2 GC.BsplineCurve
    {
        FitPoints                 = {p2};
        SymbolXY                  = {98, 110};
    }


transaction modelBased "Graph changed by user"
{
    feature bsplineSurface02 GC.BsplineSurface
    {
        Curves                    =
{bsplineCurve01[0],c1a[0],bsplineCurve12[0],c2[0]};
        Order                     = 3;
        UcurveDisplay             = 20;
        VcurveDisplay             = 20;
    }
}
```



3a - GenerativeComponents Beta
File Features Toggle Graph Tools Debug
baseCS.XYplane
Active update method
□ Show settings properties
Update Method
ByCartesianCoordinates

Property	Expression
CoordinateSystem:...	baseCS
Xtranslation: double...	point12[0].Xtranslation...
Ytranslation: double...	point12.Ytranslation ...
Ztranslation: double...	point12.Ztranslation...
Origin: IPoint (repl.)	null
Success: bool	true
X: double	{100.970, 100.970,...
XYZ: double[]	{{100.970, 78.935,...
Y: double	{78.935, 78.779,...
Z: double	{12.000, 15.000,...

Typ Point
Update m ByCartesianCoordinates
Name of featu point13

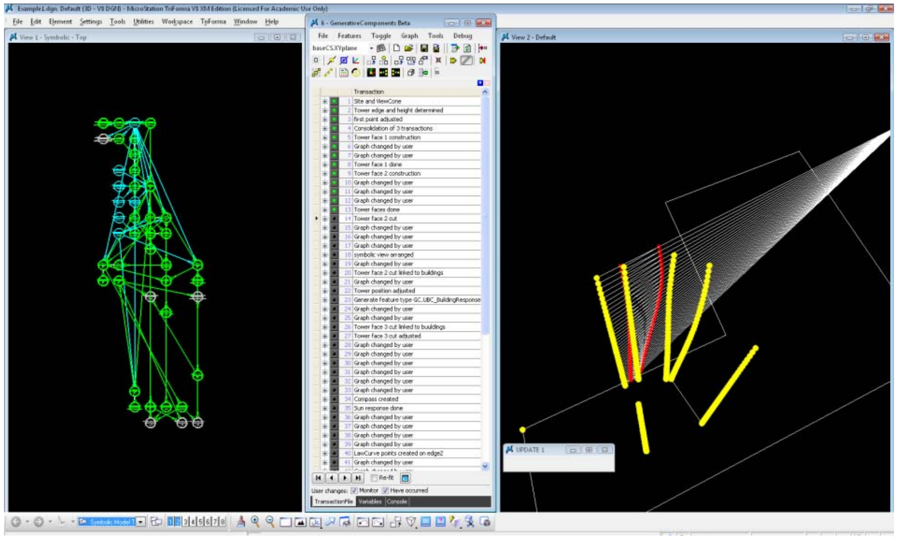
Transaction
1 Site and ViewCone
2 Tower edge and height determined
3 first point adjusted
4 Consolidation of 3 transactions
5 Tower face 1 construction
6 Graph changed by user
7 Graph changed by user
8 Tower face 1 done
9 Tower face 2 construction
10 Graph changed by user
11 Graph changed by user
12 Graph changed by user
13 Tower faces done
14 Tower face 2 cut
15 Graph changed by user
16 Graph changed by user
17 Graph changed by user
18 symbolic view arranged
19 Graph changed by user
20 Tower face 2 cut linked to buildings
21 Graph changed by user
22 Tower position adjusted
23 Generate feature type GC.UBC_BuildingResponse
24 Graph changed by user
25 Graph changed by user
26 Tower face 3 cut linked to buuldings
27 Tower face 3 cut adjusted
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Graph changed by user
34 Compass created
35 Sun response done
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 Graph changed by user
40 LawCurve points created on edge2
41 Graph changed by user

□ Re-fit
User changes: ☑ Monitor □ Have occurred

4.14

P2 and P3 (second and third corner point series) are constructed with a positing relating to P1 (first corner point series) and the side dimension. The Isocurve along the second face is determined by its relative distance between P2 and P3.

```
transaction modelBased "Graph changed by user"
{
    feature bsplineSurface01 GC.BsplineSurface
    {
        Curves                     = {c3[0],bsplineCurve14[0]};
        Order                      = 3;
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
        SymbolXY                   = {104, 111};
    }
    feature bsplineSurface04 GC.BsplineSurface
    {
        Curves                     = {bsplineCurve01[0],c4[0]};
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
        SymbolXY                   = {102, 111};
    }
    feature c3 GC.BsplineCurve
    {
        FitPoints                  = {p3};
        SymbolXY                   = {101, 108};
    }
    feature c4 GC.BsplineCurve
    {
        FitPoints                  = {p4};
        SymbolXY                   = {99, 110};
    }
    feature p4 GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = p1.X;
        Ytranslation               = p1.Y-side;
        Ztranslation               = p1.Z;
        SymbolXY                   = {99, 109};
    }
    feature s_cd GC.BsplineSurface
    {
        Curves                     = {c3[0],c4[0]};
        Order                      = 3;
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
        SymbolXY                   = {101, 111};
    }
}

transaction modelBased "Tower faces done"
{
    deleteFeature bsplineCurve01;
    deleteFeature bsplineSurface04;

    feature bsplineCurve08 GC.BsplineCurve
    {
        FitPoints                  = {p4};
    }
    feature bsplineSurface01 GC.BsplineSurface
    {
        Curves                     = {c3[0],c4[0]};
    }
    feature c1 GC.BsplineCurve
    {
        FitPoints                  = {p1};
    }
    feature c1b GC.BsplineCurve
    {
        FitPoints                  = {p1b};
    }
    feature c222222 GC.BsplineCurve
    {
        FitPoints                  = {p2};
    }
    feature s_ab GC.BsplineSurface
    {
        Curves                     =
{c1[0],c1a[0],c1b[0],c222222[0]};
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
    }
    feature s_bc GC.BsplineSurface
    {
        Curves                     = {c222222[0],c3[0]};
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
    }
```


Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug
baseCS.XYplane
View 2 - Default
UPDATE 1
Re-fit
TransactionFile Variables Console

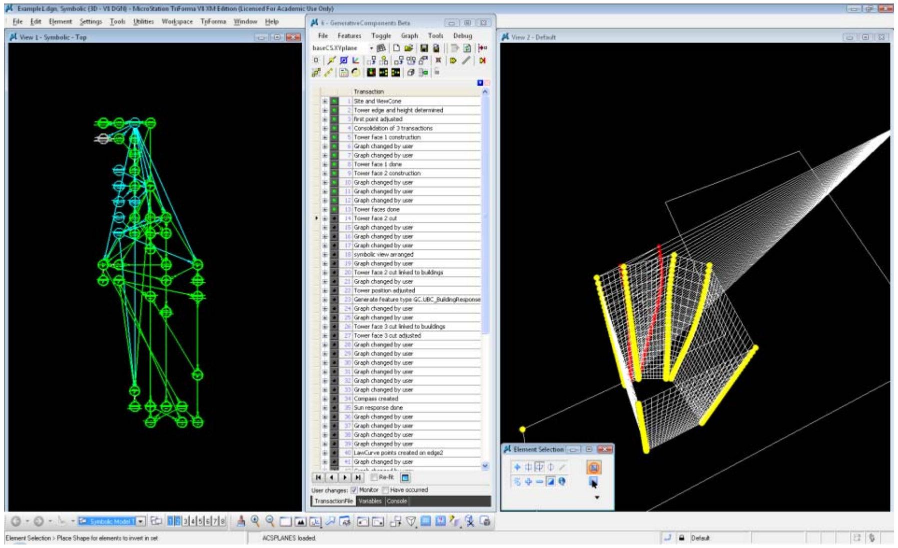
	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Graph changed by user
8	Tower face 1 done
9	Tower face 2 construction
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	Tower faces done
14	Tower face 2 cut
15	Graph changed by user
16	Graph changed by user
17	Graph changed by user
18	symbolic view arranged
19	Graph changed by user
20	Tower face 2 cut linked to buildings
21	Graph changed by user
22	Tower position adjusted
23	Generate feature type GC.UBC_BuildingResponse
24	Graph changed by user
25	Graph changed by user
26	Tower face 3 cut linked to buuldings
27	Tower face 3 cut adjusted
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Graph changed by user
34	Compass created
35	Sun response done
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	Graph changed by user
40	LawCurve points created on edge2
41	Graph changed by user

User changes: ☑ Monitor ☑ Have occurred

4.15

The third corner p3 is positioned at (side,-side) relative to the first corner p1.

The fourth corner p4 is positioned at (0,-side) relative to the first corner p1.


Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug
baseCS.XYplane
View 2 - Default
Element Selection
Re-fit
TransactionFile Variables Console
Default

	Transaction
1	Site and ViewCone
2	Tower edge and height determined
3	first point adjusted
4	Consolidation of 3 transactions
5	Tower face 1 construction
6	Graph changed by user
7	Graph changed by user
8	Tower face 1 done
9	Tower face 2 construction
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	Tower faces done
14	Tower face 2 cut
15	Graph changed by user
16	Graph changed by user
17	Graph changed by user
18	symbolic view arranged
19	Graph changed by user
20	Tower face 2 cut linked to buildings
21	Graph changed by user
22	Tower position adjusted
23	Generate feature type GC.UBC_BuildingResponse
24	Graph changed by user
25	Graph changed by user
26	Tower face 3 cut linked to buuldings
27	Tower face 3 cut adjusted
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Graph changed by user
34	Compass created
35	Sun response done
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	Graph changed by user
40	LawCurve points created on edge2
41	Graph changed by user

User changes: ☑ Monitor ☐ Have occurred

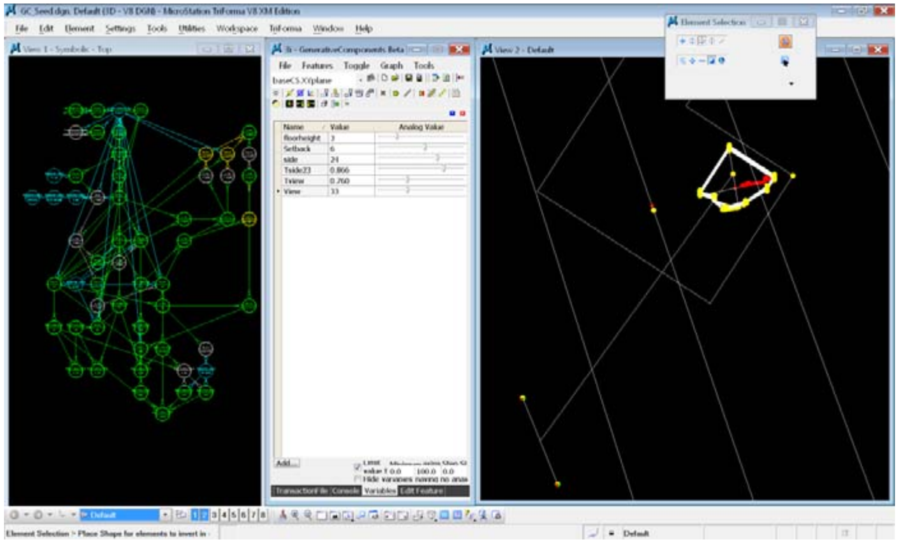
Element Selection > Place Shape for elements to invert in set

ACSPLANES loaded.

4.16

Surfaces {s_ab, s_bc, s_cd, s_da} are lofted between the edges to skin the four faces of the tower.

```
transaction modelBased "Tower face 2 cut"
{
    feature Setback GC.GraphVariable
    {
        Value                      = 4.0;
        RangeMaximum               = 10.0;
    }
    feature c2a GC.BsplineCurve
    {
        Surface                    = s_bc;
        Parameter                  = .4;
        Direction                  = DirectionOption.V;
        SymbolXY                   = {103, 114};
    }
    feature c2b GC.BsplineCurve
    {
        FitPoints                  = {p2b};
    }
    feature c2c GC.BsplineCurve
    {
        Surface                    = s_bc;
        Parameter                  = .8;
        Direction                  = DirectionOption.V;
        SymbolXY                   = {104, 114};
    }

    feature p2b GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = point16.X;
        Ytranslation               = point16.Y;
        Ztranslation               = p3.Z;
    }
    feature point16 GC.Point
    {
        Curve                      = l3;
        T                          = <free> (0.446836411407735);
        HandleDisplay              = DisplayOption.Display;
    }
    feature s_bc2 GC.BsplineSurface
    {
        Curves                     =
{c3[0],c2c,c2b[0],c2a,c222222[0]}};
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
    }
}

transaction modelBased "Graph changed by user"
{
    feature point16 GC.Point
    {
        T                          = .3;
    }
}

    feature s_da GC.BsplineSurface
    {
        Curves                     = {c1[0],c4[0]};
        UcurveDisplay              = 20;
        VcurveDisplay              = 20;
    }
}
```



Name	Value	Analog Value
floorheight	3	
Setback	6	
side	24	
Tside23	0.866	
Tview	0.760	
View	33	

4.17

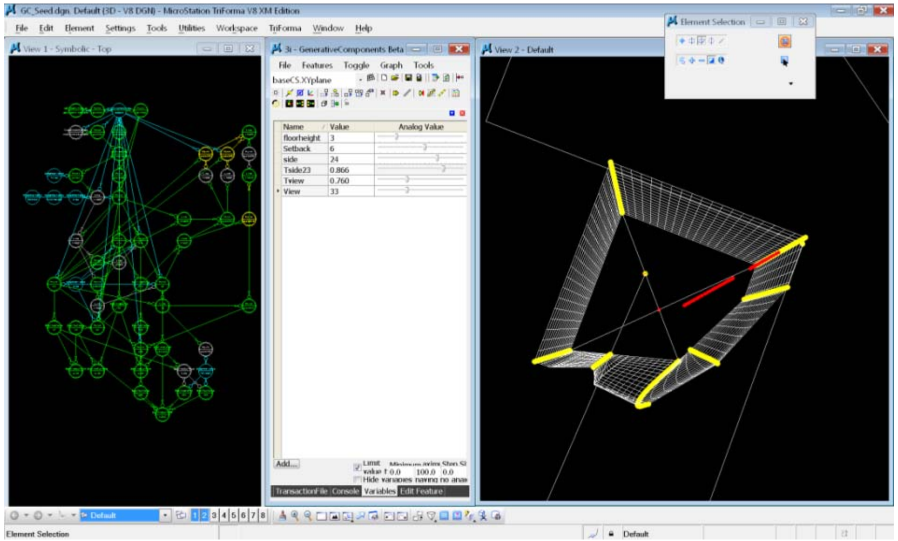
Response to buildings:

Rule: Tower should respond to the spaces between surrounding buildings to which there is a possibility for open views. A V shape is cut on the tower's face facing the in-between space, to maximise the exposed area facing the open view.

Points: building1 and building2 are created referencing the corners of two existing buildings to the north of the tower.

A bisector connects the mid-distance between the points with the tower's center.

```
transaction modelBased "Tower face 2 cut linked to buildings"
{
    feature Setback GC.GraphVariable
    {
        RangeMinimum              = 0.3;
        RangeMaximum              = 0.6;
        RangeStepSize             = 0.3;
    }
    feature Tside23 GC.GraphVariable
    {
        Value                     = 0.264;
        LimitValueToRange         = true;
        RangeMinimum              = 0.2;
        RangeMaximum              = 0.4;
        RangeStepSize             = 0.0;
    }
    feature building1 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (235.039156550752);
        Ytranslation              = <free> (78.1398182662286);
        Ztranslation              = p3[0].Z;
        SymbolXY                  = {104, 102};
        HandleDisplay             = DisplayOption.Display;
    }
    feature building2 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (240.123714996232);
        Ytranslation              = <free> (44.7562596406535);
        Ztranslation              = p3[0].Z;
        SymbolXY                  = {105, 102};
        HandleDisplay             = DisplayOption.Display;
    }
```

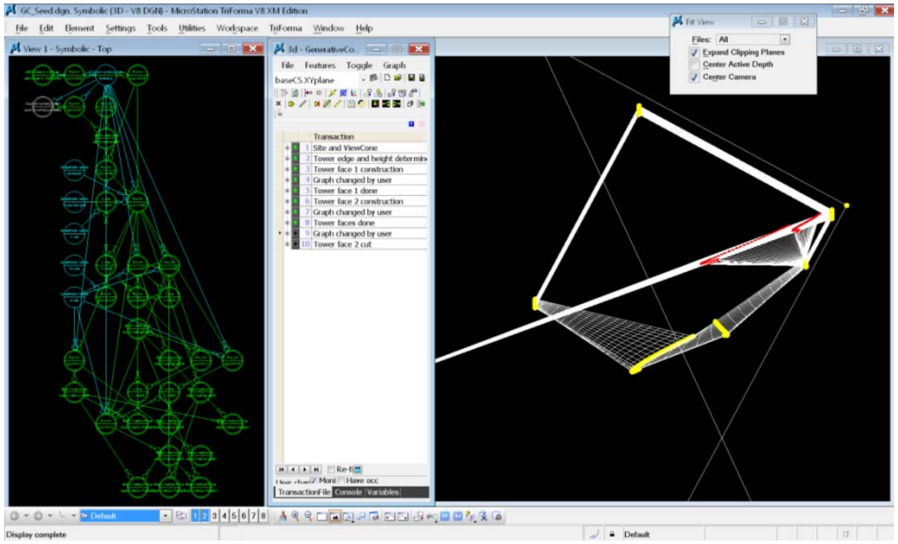

GC_Seed.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
3i - GenerativeComponents Beta
File Features Toggle Graph Tools
baseCS.XYplane
View 2 - Default
Element Selection
Default

Name	Value	Analog Value
floorheight	3	
Setback	6	
side	24	
Tside23	0.866	
Tview	0.760	
View	33	

4.18

A V shape cut is made at the intersection of the bisector and the tower's north face. The isocurve that defines the V cut leans to the inside of the tower as it rises through the floors, giving greater view exposure at the higher levels.

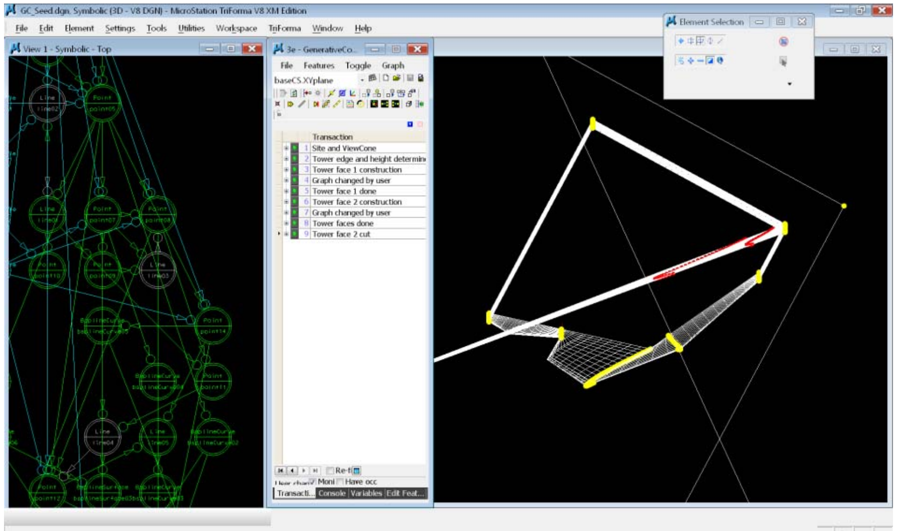
```
    feature buildingtarget GC.Point
    {
        Curve                          = line07;
        T                              = <free> (0.586405318050557);
        HandleDisplay                  = DisplayOption.Display;
    }
    feature line05 GC.Line
    {
        StartPoint                     = building1;
        EndPoint                       = building2;
        SymbolXY                       = {106, 101};
        Display                        = DisplayOption.Display;
    }
    feature line06 GC.Line
    {
        StartPoint                     = p3[0];
        EndPoint                       = p1[0];
        SymbolXY                       = {103, 105};
        Display                        = DisplayOption.Display;
    }
    feature line07 GC.Line
    {
        StartPoint                     = p2[0];
        EndPoint                       = p4[0];
        SymbolXY                       = {103, 106};
        Display                        = DisplayOption.Display;
    }
    feature line08 GC.Line
    {
        StartPoint                     = building1;
        EndPoint                       = point10;
        SymbolXY                       = {104, 103};
        Display                        = DisplayOption.Hide;
    }
    feature line09 GC.Line
    {
        StartPoint                     = building2;
        EndPoint                       = point10;
        SymbolXY                       = {105, 103};
        Display                        = DisplayOption.Hide;
    }
    feature line10 GC.Line
    {
        StartPoint                     = point08;
        EndPoint                       = buildingtarget;
        SymbolXY                       = {106, 103};
    }
    feature p2b GC.Point
    {
        Xtranslation                   = point12.X;
        Ytranslation                   = point12.Y;
        SymbolXY                       = {101, 111};
    }
    feature point08 GC.Point
    {
        Curve                          = line05;
        T                              = .5;
        SymbolXY                       = {106, 102};
        Display                        = DisplayOption.Hide;
        HandleDisplay                  = DisplayOption.Display;
    }
    feature point10 GC.Point
    {
        Intersector0                   = line06;
        Intersector1                   = line07;
        SymbolXY                       = {105, 105};
    }
    feature point12 GC.Point
    {
        Intersector0                   = line10;
        Intersector1                   = l3;
    }
    feature point16 GC.Point
    {
        T                              = Tside23;
    }
}
```


GC_Seed.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
3d - GenerativeCo...
File Features Toggle Graph
baseCS.XYplane
Fit View
Files: All
Expand Clipping Planes
Center Active Depth
Center Camera
Console Variables
Default
Display complete

	Transaction
1	Site and ViewCone
2	Tower edge and height determin
3	Tower face 1 construction
4	Graph changed by user
5	Tower face 1 done
6	Tower face 2 construction
7	Graph changed by user
8	Tower faces done
9	Graph changed by user
10	Tower face 2 cut

4.19

Top view after the V cut in response to the In-between space.

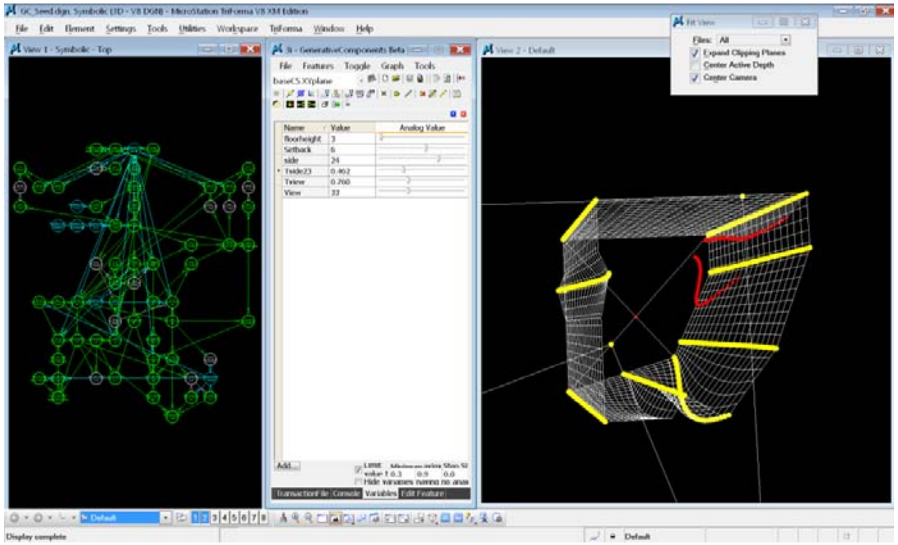

GC_Seed.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
3e - GenerativeCo...
File Features Toggle Graph
baseCS.XYplane
Element Selection
Console Variables Edit Feat...

	Transaction
1	Site and ViewCone
2	Tower edge and height determin
3	Tower face 1 construction
4	Graph changed by user
5	Tower face 1 done
6	Tower face 2 construction
7	Graph changed by user
8	Tower faces done
9	Tower face 2 cut

4.20

Top view after the V cut in response to the In-between space.

```
transaction generateFeatureType "Generate feature type
GC.UBC_BuildingResponse"
{
    type                       = GC.UBC_BuildingResponse;
    loadIntoFutureSessions     = true;
    inputProperties            = {
                                     property GC.IPoint baseCS
                                     {
                                         feature                    =
baseCS;
                                         isReplicatable             =
true;
                                         isParentModel              =
true;
                                     }
                                     property GC.IPoint building1
                                     {
                                         feature                    =
building1;
                                         isReplicatable             =
true;
                                     }
                                     property GC.IPoint building2
                                     {
                                         feature                    =
building2;
                                         isReplicatable             =
true;
                                     }
                                     property GC.IPoint
buildingtarget
                                     {
                                         feature                    =
buildingtarget;
                                         isReplicatable             =
true;
                                     }
                                     property GC.BsplineCurve
c222222
                                     {
                                         feature                    =
c222222;
                                         isOptional                 =
true;
                                         isReplicatable             =
true;
                                     }
                                     property GC.BsplineCurve c2a
                                     {
                                         feature                    =
c2a;
                                         isOptional                 =
true;
                                         isReplicatable             =
true;
                                     }
                                     property GC.BsplineCurve c2c
                                     {
                                         feature                    =
c2c;
                                         isOptional                 =
true;
                                         isReplicatable             =
true;
                                     }
                                     property GC.IPoint point10
                                     {
                                         feature                    =
point10;
                                         isOptional                 =
true;
                                         isReplicatable             =
true;
                                     }
                                 };
    outputProperties           = {
                                     property GC.BsplineCurve c2b
                                     {
                                         feature                    =
c2b;
                                         isDynamic                  =
true;
                                     }
                                     property GC.BsplineCurve c3
                                     {
```


GC_Seed.dgn: Symbolic UD - V8 DGN - MicroStation TriForma V8 XM Edition
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
GenerativeComponents Beta
File Features Toggle Graph Tools
baseCS.XYplane
View 2 - Default
Fit View
Default
Display complete

Name	Value	Analog Value
floorheight	3	
Setback	6	
side	24	
Tside23	0.462	
Tview	0.760	
View	33	

4.21

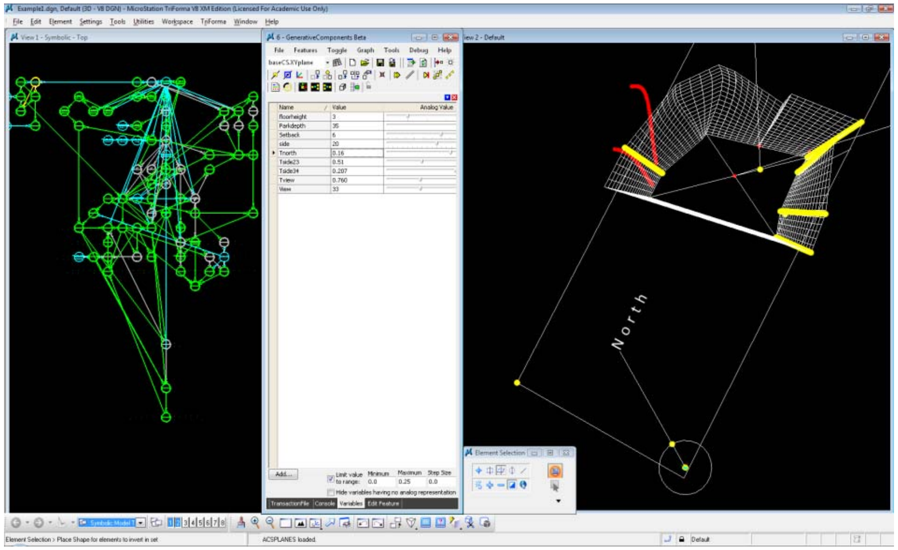
Building Response Feature generation:

A Feature for the building response reaction to In-between views UBC_BuildingResponse is generated to be part of the software interface. This way the response can be easily applied to any of the faces of the tower, given the input values of the surrounding buildings' coordinates.

The feature UBC_BuildingResponse is used to cut a similar V shape on the tower's east face, which responds to two other buildings surrounding the site. The V cut width is directly proportional to the open view width between the 2 surrounding buildings.

```
                                        feature                   =
c3;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Line l3
                                    {
                                        feature                   =
l3;
                                        isDynamic                 =
true;
                                    }

                                    property GC.Line line05
                                    {
                                        feature                   =
line05;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Line line10
                                    {
                                        feature                   =
line10;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Point p2b
                                    {
                                        feature                   =
p2b;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Point p3
                                    {
                                        feature                   =
p3;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Point point12
                                    {
                                        feature                   =
point12;
                                        isDynamic                 =
true;
                                    }
                                    property GC.BsplineSurface
s_bc2
                                    {
                                        feature                   =
s_bc2;
                                        isDynamic                 =
true;
                                    }
                                };
    internalProperties        = {
                                    property GC.Line line08
                                    {
                                        feature                   =
line08;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Line line09
                                    {
                                        feature                   =
line09;
                                        isDynamic                 =
true;
                                    }
                                    property GC.Point point08
                                    {
                                        feature                   =
point08;
                                        isDynamic                 =
true;
                                    }
                                };
}
```

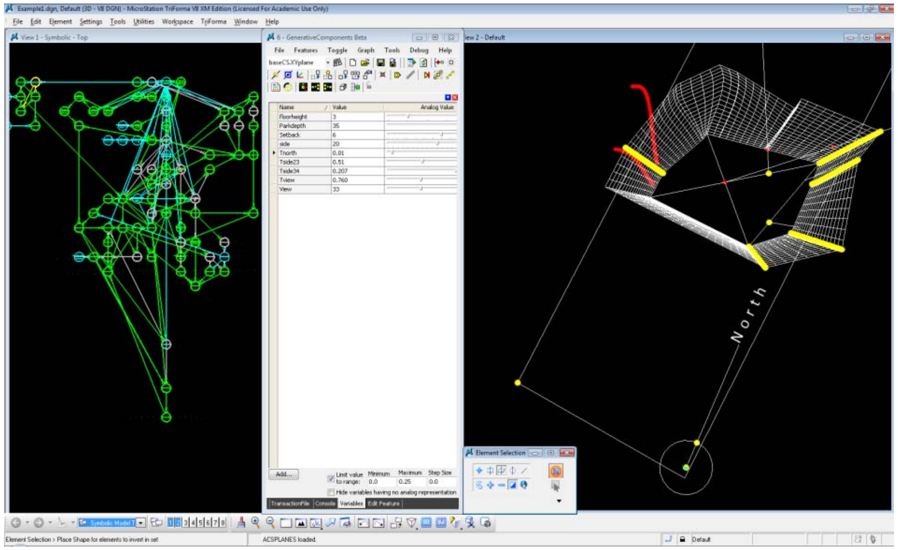
```
transaction modelBased "North"
{
    feature Tnorth GC.GraphVariable
    {
        Value                     = 0.25;
        LimitValueToRange         = true;
        RangeMaximum              = 1.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {87, 101};
    }
    feature bsplineCurve03 GC.BsplineCurve
    {
        Parameter                 = Tside34+.5;
    }
    feature circle01 GC.Circle
    {
        CenterPoint               = point20;
        Radius                    = 5;
        Support                   = baseCS.XYplane;
        SymbolXY                  = {90, 100};
    }
    feature east GC.Point
    {
        Curve                     = circle01;
        T                         = north.T-.25;
        SymbolXY                  = {90, 103};
        HandleDisplay             = DisplayOption.Display;
    }
    feature line15 GC.Line
    {
        StartPoint                = point20;
        Direction                 = northdirection;
        Length                    = circle01.Radius+2;
        SymbolXY                  = {91, 101};
    }
    feature north GC.Point
    {
        Curve                     = circle01;
        T                         = Tnorth;
        SymbolXY                  = {90, 101};
        HandleDisplay             = DisplayOption.Display;
    }
    feature northdirection GC.Direction
    {
        OriginPoint               = point20;
        DirectionPoint            = north;
        SymbolXY                  = {90, 102};
    }
    feature text01 GC.Text
    {
        Placement                 = line15.EndPoint;
        TextString                = " North ";
        SymbolXY                  = {91, 103};
    }
}

transaction modelBased "Compass created"
{
    feature direction01 GC.Direction
    {
        OriginPoint               = point20;
        DirectionPoint            = east;
        SymbolXY                  = {90, 104};
    }
}
transaction modelBased "Sun response done"
{
    feature Tnorth GC.GraphVariable
    {
        Value                     = 0.165;
        RangeMaximum              = 0.25;
        SymbolXY                  = {89, 101};
    }
```



Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
North
Element Selection
TransactionFile Console Variables Edit Feature

Name	Value	Analog Value
floorheight	3	
Parkdepth	35	
Setback	6	
side	20	
Tnorth	0.16	
Tside23	0.51	
Tside34	0.207	
Tview	0.760	
View	33	

4.22



Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
North
Element Selection
TransactionFile Console Variables Edit Feature

Name	Value	Analog Value
floorheight	3	
Parkdepth	35	
Setback	6	
side	20	
Tnorth	0.01	
Tside23	0.51	
Tside34	0.207	
Tview	0.760	
View	33	

4.23

```
feature Tside23 GC.GraphVariable
    {
        Value                     = 0.51;
    }
    feature line15 GC.Line
    {
        Length                    = circle01.Radius+70;
    }
    feature p3 GC.Point
    {
        Ytranslation              = p1.Y-side-Abs(north.X);
    }
    feature p4 GC.Point
    {
        Xtranslation              = p1.X+Abs(north.Y);
        Ytranslation              = p1.Y-side-Abs(north.Y);
    }
    feature point03 GC.Point
    {
        Display                   = DisplayOption.Hide;
    }
    feature point20 GC.Point
    {
        Xtranslation              = point03.X;
        Ytranslation              = point03.Y;
    }
}
```

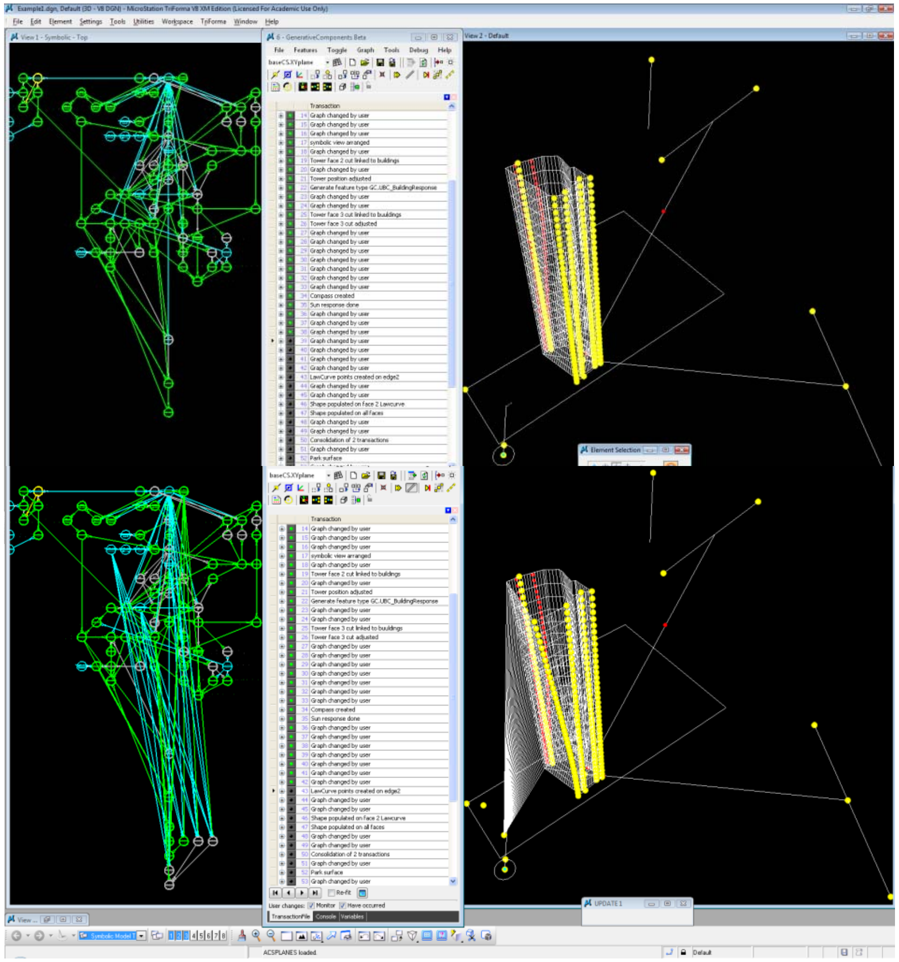
Response to South exposure:

Rule: The tower should respond to the North direction in order to maximise the southern exposure.

Tnorth is a variable that controls the North direction relative to the site. It works by defining the position of the North point relative to its displacement over the circle (compass).

The X and Y translation of Edges p3 and p4 are linked to the North direction. The translation changes relative to the original position to allow more south exposure on the building's faces. The first screenshot shows the real North direction relative to the site, while both screenshots together show the edges' response to a change in the North direction (or a change in the tower's orientation). (Figures 4.22, 4.23)

```
transaction modelBased "Response to park"
{
    feature line17 GC.Line
    {
        StartPoint                 = point33;
        EndPoint                   = p1;
    }
    feature p4 GC.Point
    {
        Xtranslation               =
p1.X+Abs(north.Y)+line17[0].Length*.1-line17.Length*.1;
    }
    feature point32 GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = point04.X+Setback;
        Ytranslation               = point04.Y-Setback;
        Ztranslation               = point04.Z;
        HandleDisplay              = DisplayOption.Display;
    }
    feature point33 GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = point04.X+Setback;
        Ytranslation               = point04.Y-Setback-side;
        Ztranslation               = point04.Z;
    }
    feature point34 GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = point04.X+3*Setback;
        Ytranslation               = point04.Y-Setback-side;
        Ztranslation               = point04.Z+2;
    }
    feature point35 GC.Point
    {
        CoordinateSystem           = baseCS;
        Xtranslation               = point04.X+6*Setback;
        Ytranslation               = point04.Y-Setback-side;
        Ztranslation               = point04.Z+1;
    }
}
```



```
Transaction
14 Graph changed by user
15 Graph changed by user
16 Graph changed by user
17 symbolic view arranged
18 Graph changed by user
19 Tower face 2 cut linked to buildings
20 Graph changed by user
21 Tower position adjusted
22 Generate feature type GC.UBC_BuildingResponse
23 Graph changed by user
24 Graph changed by user
25 Tower face 3 cut linked to buildings
26 Tower face 3 cut adjusted
27 Graph changed by user
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Graph changed by user
34 Compass created
35 Sun response done
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 Graph changed by user
40 Graph changed by user
41 Graph changed by user
42 Graph changed by user
43 LawCurve points created on edge2
44 Graph changed by user
45 Graph changed by user
46 Shape populated on face 2 Lawcurve
47 Shape populated on all faces
48 Graph changed by user
49 Graph changed by user
50 Consolidation of 2 transactions
51 Graph changed by user
52 Park surface
53 Graph changed by user
```

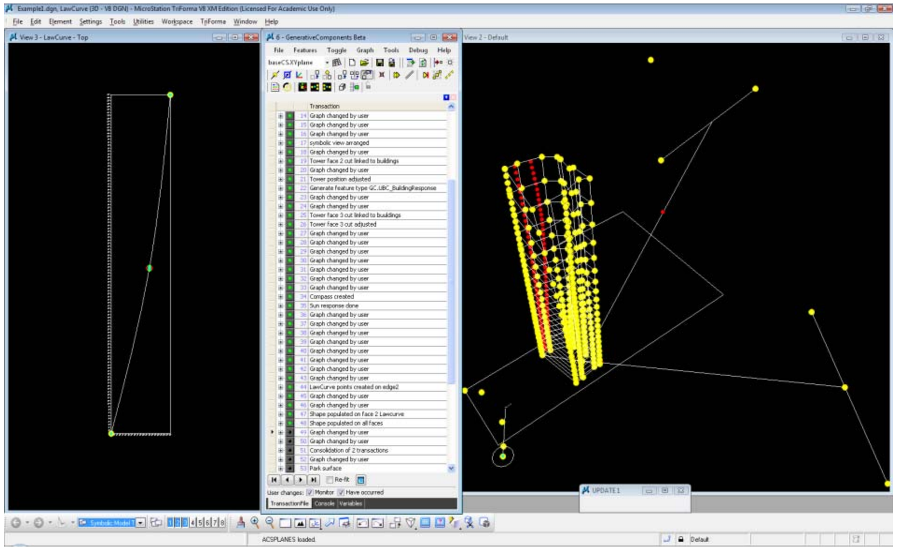
4.24

Response to Park:

Rule: The park influences the side of the tower facing it by maximising the units' exposure towards the site of the park.

Corner point series p4's x translation is adjusted to reflect the relationship with the park's site. A point `point33` is created at the park's edge, and linked with line `line17` to the tower's face. The X translation of `p4` is directly influenced by the linking line's length.

```
transaction modelBased "LawCurve points created on edge2"
{
    feature LawC1 GC.Point
    {
        CoordinateSystem          = coordinateSystem02;
        Xtranslation              = 0;
        Ytranslation              = 0;
        Ztranslation              = <free> (0.0);
        HandleDisplay             = DisplayOption.Display;
    }
    feature LawC2 GC.Point
    {
        CoordinateSystem          = coordinateSystem02;
        Xtranslation              = <free> (9.6936374195728);
        Ytranslation              = <free> (37.6503328615144);
        Ztranslation              = <free> (0.0);
        HandleDisplay             = DisplayOption.Display;
    }
    feature LawC3 GC.Point
    {
        CoordinateSystem          = coordinateSystem02;
        Xtranslation              = lawCurveFrame01.Xdimension;
        Ytranslation              = lawCurveFrame01.Ydimension;
        Ztranslation              = <free> (0.0);
        HandleDisplay             = DisplayOption.Display;
    }
    feature bsplineCurve05 GC.BsplineCurve
    {
        FitPoints                 = {point05,point07,building1};
        SymbolXY                  = {89, 107};
    }
    feature coordinateSystem02 GC.CoordinateSystem
    {
        Model                     = "LawCurve";
    }
    feature lawCurve01 GC.LawCurve
    {
        LawCurveFrame             = lawCurveFrame01;
        ControlPoints             = {LawC1,LawC2,LawC3};
        CurveControl              = CurveOption.ByPoints;
        Order                     = 4;
        IndependentVariable       = Series(0,20,1);
    }
    feature lawCurveFrame01 GC.LawCurveFrame
    {
        Plane                     = coordinateSystem02.XYplane;
        Xdimension                = 20;
        Ydimension                = c222222[0].Length;
        Xaxis                     = Series(0,20,1);
        Yaxis                     = Series(0,c222222[0].Length,1);
    }
    feature p2_new GC.Point
    {
        Curve                     = c222222[0];
        DistanceAlongCurve        = lawCurve01.DependentVariable;
    }
}
```


Example1.dgn, LawCurve (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 3 - LawCurve - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
View 2 - Default
Re-fit
TransactionFile Console Variables
UPDATE1
Default

Transaction
14 Graph changed by user
15 Graph changed by user
16 Graph changed by user
17 symbolic view arranged
18 Graph changed by user
19 Tower face 2 cut linked to buildings
20 Graph changed by user
21 Tower position adjusted
22 Generate feature type GC.UBC_BuildingResponse
23 Graph changed by user
24 Graph changed by user
25 Tower face 3 cut linked to buildings
26 Tower face 3 cut adjusted
27 Graph changed by user
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Graph changed by user
34 Compass created
35 Sun response done
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 Graph changed by user
40 Graph changed by user
41 Graph changed by user
42 Graph changed by user
43 Graph changed by user
44 LawCurve points created on edge2
45 Graph changed by user
46 Graph changed by user
47 Shape populated on face 2 Lawcurve
48 Shape populated on all faces
49 Graph changed by user
50 Graph changed by user
51 Consolidation of 2 transactions
52 Graph changed by user
53 Park surface

User changes: ☑ Monitor ☑ Have occurred

ACSPLANES loaded

4.25

Law Curve:

Rule: Glazing to solid ratio should increase towards the higher levels of the building to make use of view exposures. At the same instance, Openings at the base levels should be bigger to compensate for the decrease in lighting conditions.

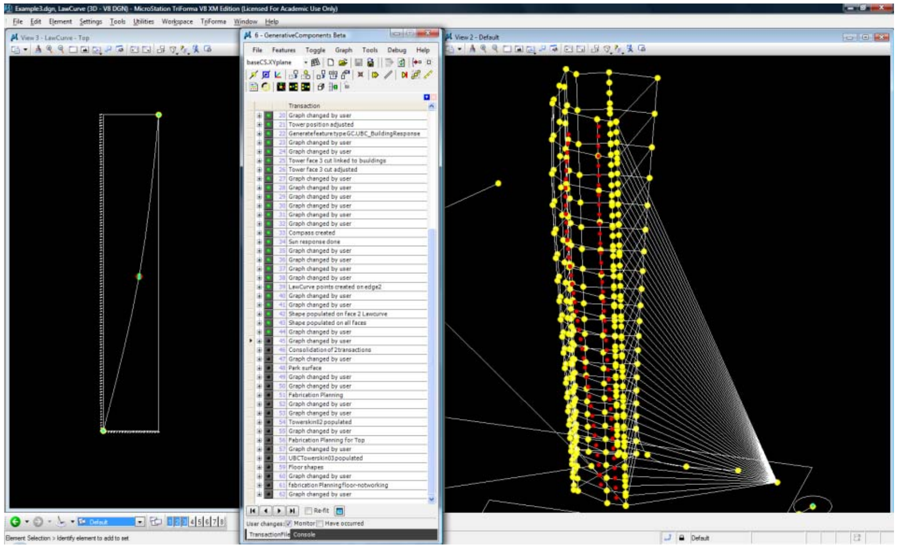
A law curve is applied to control the spacing of points along the tower edges. The law curve is set on a graph in which:

1. The X-axis (independent) represents the number of points along the curve.
2. The Y-axis (dependent) represents the spacing between the points.

The law curve has control points {LawC1, Lawc2, Lawc3} that changes its curvature, thus changes the gradient spacing of points along the curve (Y).

Example: if the curve is a straight line, the spacing between the points (Y) is equal.

```
transaction modelBased "LawCurve points created on edge2"
{
    feature LawC1 GC.Point
    {
        CoordinateSystem            = coordinateSystem02;
        Xtranslation                = 0;
        Ytranslation                = 0;
        Ztranslation                = <free> (0.0);
        HandleDisplay               = DisplayOption.Display;
    }
    feature LawC2 GC.Point
    {
        CoordinateSystem            = coordinateSystem02;
        Xtranslation                = <free> (9.6936374195728);
        Ytranslation                = <free> (37.6503328615144);
        Ztranslation                = <free> (0.0);
        HandleDisplay               = DisplayOption.Display;
    }
    feature LawC3 GC.Point
    {
        CoordinateSystem            = coordinateSystem02;
        Xtranslation                = lawCurveFrame01.Xdimension;
        Ytranslation                = lawCurveFrame01.Ydimension;
        Ztranslation                = <free> (0.0);
        HandleDisplay               = DisplayOption.Display;
    }
    feature bsplineCurve05 GC.BsplineCurve
    {
        FitPoints                   = {point05,point07,building1};
        SymbolXY                    = {89, 107};
    }
    feature coordinateSystem02 GC.CoordinateSystem
    {
        Model                       = "LawCurve";
    }
    feature lawCurve01 GC.LawCurve
    {
        LawCurveFrame               = lawCurveFrame01;
        ControlPoints               = {LawC1,LawC2,LawC3};
        CurveControl                = CurveOption.ByPoints;
        Order                       = 4;
        IndependentVariable         = Series(0,20,1);
    }
    feature lawCurveFrame01 GC.LawCurveFrame
    {
        Plane                       = coordinateSystem02.XYplane;
        Xdimension                  = 20;
        Ydimension                  = c222222[0].Length;
        Xaxis                       = Series(0,20,1);
        Yaxis                       = Series(0,c222222[0].Length,1);
    }
    feature p2_new GC.Point
    {
        Curve                       = c222222[0];
        DistanceAlongCurve          = lawCurve01.DependentVariable;
    }
}
```



Transaction
20 Graph changed by user
21 Tower position adjusted
22 Generatefeaturetype GC.UBC_BuildingResponse
23 Graph changed by user
24 Graph changed by user
25 Tower face 3 cut linked to buildings
26 Tower face 3 cut adjusted
27 Graph changed by user
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Compass created
34 Sun response done
35 Graph changed by user
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 LawCurve points created on edge2
40 Graph changed by user
41 Graph changed by user
42 Shape populated on face 2 Lawcurve
43 Shape populated on all faces
44 Graph changed by user
45 Graph changed by user
46 Consolidation of 2transactions
47 Graph changed by user
48 Park surface
49 Graph changed by user
50 Graph changed by user
51 Fabrication Planning
52 Graph changed by user
53 Graph changed by user
54 Towerskin02 populated
55 Graph changed by user
56 Fabrication Planning for Top
57 Graph changed by user
58 UBCTowerskin03 populated
59 Floor shapes
60 Graph changed by user
61 fabrication Planningfloor-notworking
62 Graph changed by user

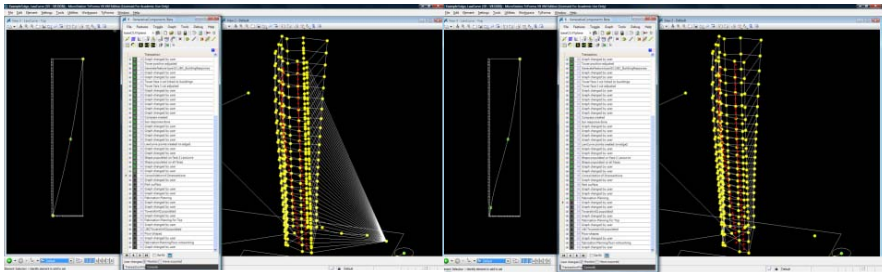
User changes: ☑ Monitor □ Have occurred

Element Selection > Identify element to add to set

4.26

Points are created on the tower's edges and isocurves with reference to the law curve's Y axis (spacing between instances of each point).

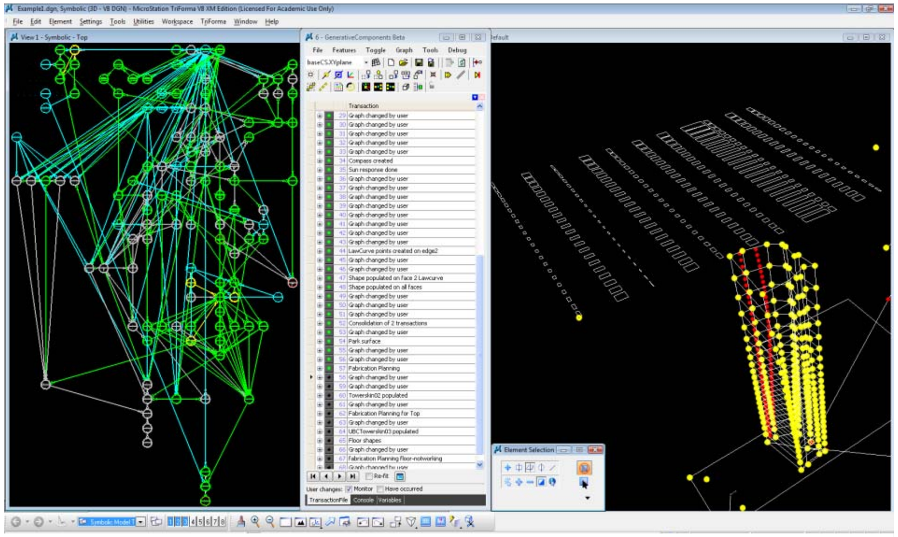
A rectangular polygon `shape01` is created on top of these points by joining them. The polygons can be used as a base to populate skin design components to the tower's surface, while responding to the various dimensions of each individual polygon.



4.27

Adjustments done to the Law Curve's control point reflects in a change in Y translation of the polygon points. The Law Curve is now manipulated by four control points to change the points' spacing at the top and bottom levels of the tower.

```
transaction modelBased "Fabrication Planning"
{

    feature coordinateSystem03 GC.CoordinateSystem
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = 0;
        Ytranslation              = 200;
        Ztranslation              = 0;
    }
    feature export01 GC.Export
    {
        FeaturesToExport          = fabricationPlanning01;
        ExportDesignFile          = 1;
        ExportSeedFile            = 2;
    }
    feature fabricationPlanning01 GC.FabricationPlanning
    {
        CoordinateSystem          = coordinateSystem03;
        Shapes                    = shape01;
        Xspacing                  = 20;
        Yspacing                  = 1;
        Fill                      = true;
        TextStyle                 = textStyle01;
    }
}
```



Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug
baseCS.XYplane
Default
Element Selection
Symbolic Model

	Transaction
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Graph changed by user
34	Compass created
35	Sun response done
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	Graph changed by user
40	Graph changed by user
41	Graph changed by user
42	Graph changed by user
43	Graph changed by user
44	LawCurve points created on edge2
45	Graph changed by user
46	Graph changed by user
47	Shape populated on face 2 Lawcurve
48	Shape populated on all faces
49	Graph changed by user
50	Graph changed by user
51	Graph changed by user
52	Consolidation of 2 transactions
53	Graph changed by user
54	Park surface
55	Graph changed by user
56	Graph changed by user
57	Fabrication Planning
58	Graph changed by user
59	Graph changed by user
60	Towerskin02 populated
61	Graph changed by user
62	Fabrication Planning for Top
63	Graph changed by user
64	UBCTowerskin03 populated
65	Floor shapes
66	Graph changed by user
67	Fabrication Planning floor-notworking
68	Graph changed by user

☐ Re-fit

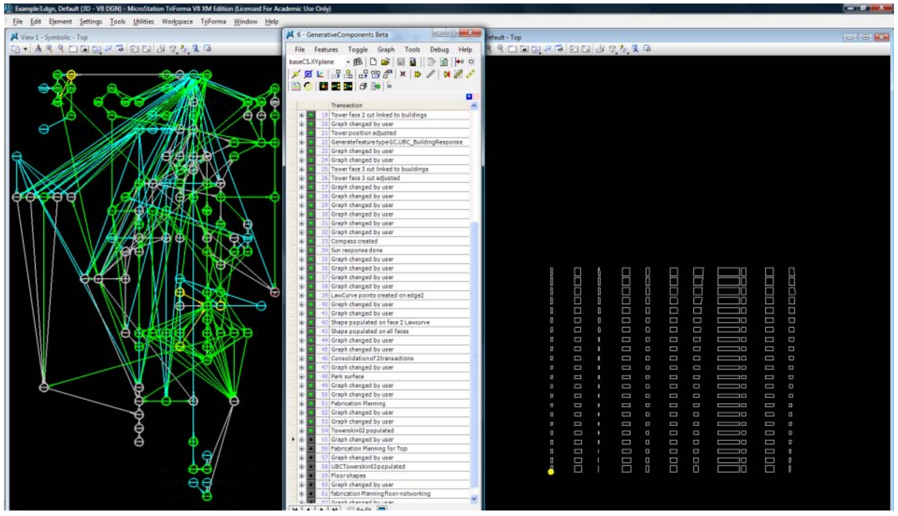
User changes: ☑ Monitor ☐ Have occurred

TransactionFile | Console | Variables

4.28

Fabrication Planning:

Fabrication Planning is a built in feature that can prepare geometry for digitalfabrication. `Shape01` (the tower's skin) with its varying instances is laid flat on a XY plane. The flat polygon layout is dynamically linked to changes introduced to the tower's geometry.



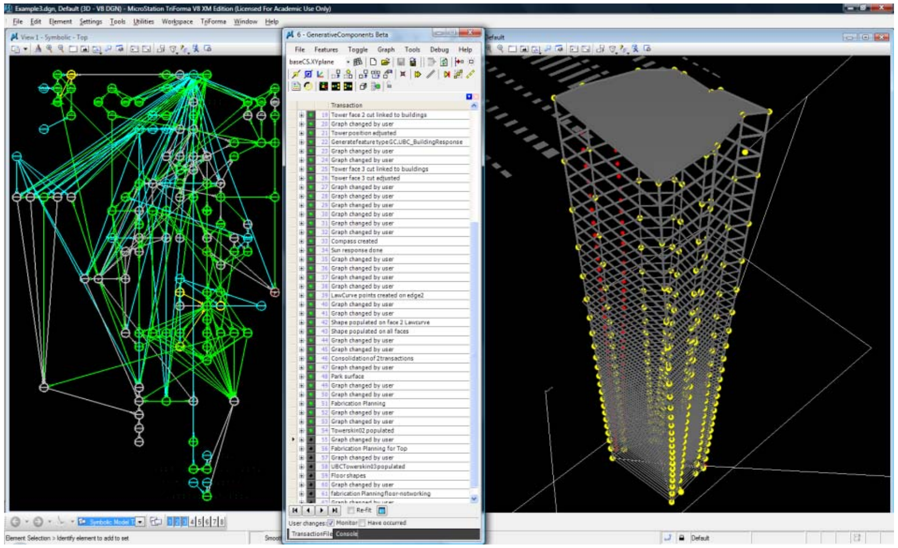
Example3.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
Default - Top

	Transaction
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generatefeature typeGC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buuldings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user
41	Graph changed by user
42	Shape populated on face 2 Lawcurve
43	Shape populated on all faces
44	Graph changed by user
45	Graph changed by user
46	Consolidationof 2transactions
47	Graph changed by user
48	Park surface
49	Graph changed by user
50	Graph changed by user
51	Fabrication Planning
52	Graph changed by user
53	Graph changed by user
54	Towerskin02 populated
55	Graph changed by user
56	Fabrication Planning for Top
57	Graph changed by user
58	UBCTowerskin03populated
59	Floor shapes
60	Graph changed by user
61	fabrication Planningfloor-notworking
62	Graph changed by user

4.29

```
transaction modelBased "Towerskin02 populated"
{
    feature Rcomponent GC.GraphVariable
    {
        Value                      = .2;
        LimitValueToRange          = true;
        RangeMinimum               = 0.1;
        RangeMaximum               = 0.6;
        RangeStepSize              = 0.1;
    }
    feature Tnumber GC.GraphVariable
    {
        Value                      = .333333;
    }
    feature p4 GC.Point
    {
        Display                    = DisplayOption.Hide;
    }
    feature shape01 GC.Shape
    {
        Display                    = DisplayOption.Hide;
    }
    feature uBCTowerskin0201 GC.UBCTowerskin02
    {
        Rcomponent                 = Rcomponent;
        Tnumber                    = Tnumber;
        shape01                    = shape01;
    }
    feature uBCViewCone01 GC.UBCViewCone
    {
        Display                    = DisplayOption.Hide;
    }
}
```

```
transaction modelBased "UBCTowerskin03 populated"
{
    feature uBCTowerskin0301 GC.UBCTowerskin03
    {
        Rcomponent                 = Rcomponent;
        Tnumber                    = Tnumber;
        shape01                    = shape01;
        SymbolXY                   = {103, 126};
        Display                    = DisplayOption.Display;
    }
}
```

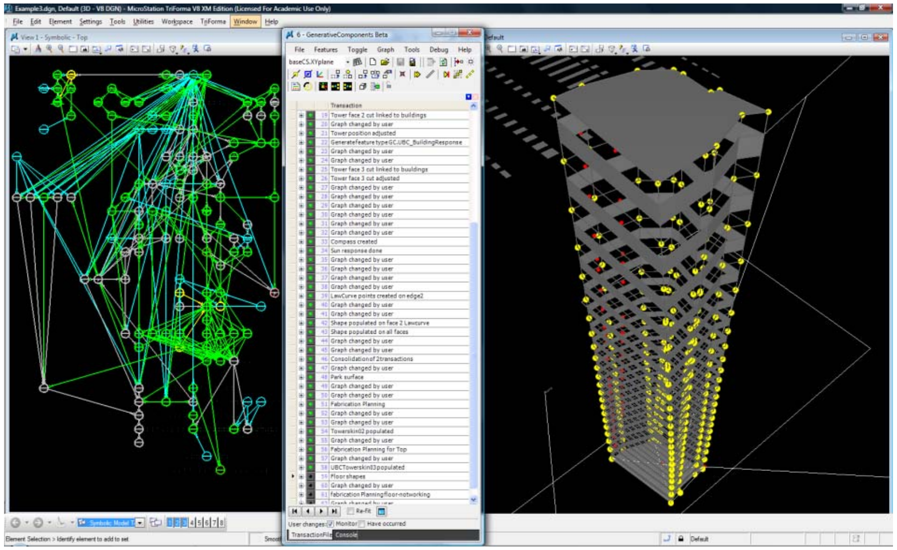

Example3.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
Default
Re-Fit
User changes: Monitor Have occurred
TransactionFile Console
Element Selection > Identify element to add to set

	Transaction
19	Tower face 2 cut linked to buildings
20	Graph changed by user
21	Tower position adjusted
22	Generatefeature typeGC.UBC_BuildingResponse
23	Graph changed by user
24	Graph changed by user
25	Tower face 3 cut linked to buuldings
26	Tower face 3 cut adjusted
27	Graph changed by user
28	Graph changed by user
29	Graph changed by user
30	Graph changed by user
31	Graph changed by user
32	Graph changed by user
33	Compass created
34	Sun response done
35	Graph changed by user
36	Graph changed by user
37	Graph changed by user
38	Graph changed by user
39	LawCurve points created on edge2
40	Graph changed by user
41	Graph changed by user
42	Shape populated on face 2 Lawcurve
43	Shape populated on all faces
44	Graph changed by user
45	Graph changed by user
46	Consolidationof 2transactions
47	Graph changed by user
48	Park surface
49	Graph changed by user
50	Graph changed by user
51	Fabrication Planning
52	Graph changed by user
53	Graph changed by user
54	Towerskin02 populated
55	Graph changed by user
56	Fabrication Planning for Top
57	Graph changed by user
58	UBCTowerskin03populated
59	Floor shapes
60	Graph changed by user
61	fabrication Planningfloor-notworking

4.30

Polygon Shape01 is used to populate facade components across the building. The varying spacing of the polygon points allows for varying stretching of the component over each of the polygon's instance. In this case, the component divides each polygon into 3 horizontal panels, two of which will be glazed. This translates into different glazing sizes across the tower's levels, matching the different glazing to solid ratios rule.

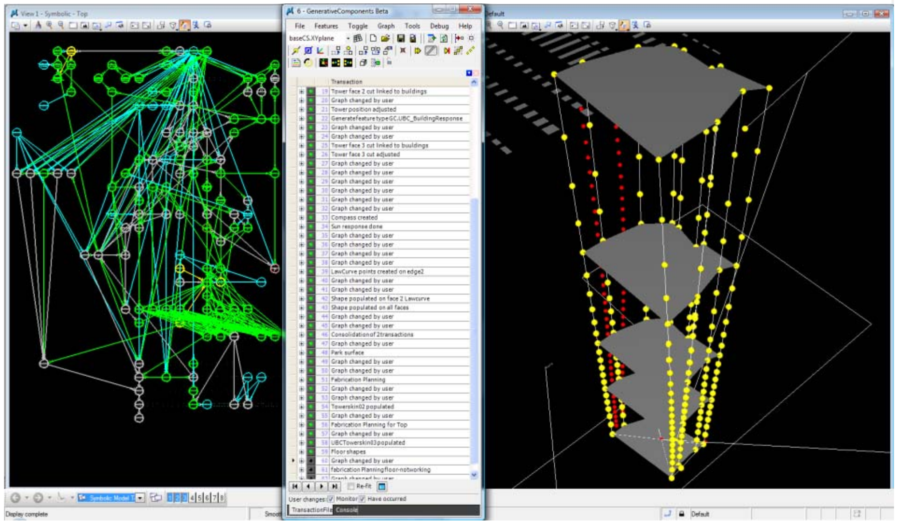
```
transaction modelBased "Floor shapes"
{
    deleteFeature fabricationPlanning03;
    feature shape05 GC.Shape
    {
        Vertices                  =
{point27[18],point28[18],point29[18],point23[18],point21[18],point2
2[18],point19[18],p2_new[18],point24[18],point25[18],point26[18]};
        SymbolXY                  = {105, 121};
    }
    feature shape06 GC.Shape
    {
        Vertices                  =
{point27[12],point28[12],point29[12],point23[12],point21[12],point2
2[12],point19[12],p2_new[12],point24[12],point25[12],point26[12]};
        SymbolXY                  = {106, 121};
    }
    feature shape07 GC.Shape
    {
        Vertices                  =
{point27[6],point28[6],point29[6],point23[6],point21[6],point22[6],
point19[6],p2_new[6],point24[6],point25[6],point26[6]};
        SymbolXY                  = {107, 121};
    }
    feature textStyle01 GC.TextStyle
    {
        SymbolXY                  = {101, 126};
    }
}

transaction modelBased "Graph changed by user"
{
    feature shape06 GC.Shape
    {
        Vertices                  =
{point27[14],point28[14],point29[14],point23[14],point21[14],point2
2[14],point19[14],p2_new[14],point24[14],point25[14],point26[14]};
    }
}
```


Example3.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
Default

Transaction
19 Tower face 2 cut linked to buildings
20 Graph changed by user
21 Tower position adjusted
22 Generatefeature type GC.UBC_BuildingResponse
23 Graph changed by user
24 Graph changed by user
25 Tower face 3 cut linked to buuldings
26 Tower face 3 cut adjusted
27 Graph changed by user
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Compass created
34 Sun response done
35 Graph changed by user
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 LawCurve points created on edge2
40 Graph changed by user
41 Graph changed by user
42 Shape populated on face 2 Lawcurve
43 Shape populated on all faces
44 Graph changed by user
45 Graph changed by user
46 Consolidation of 2transactions
47 Graph changed by user
48 Park surface
49 Graph changed by user
50 Graph changed by user
51 Fabrication Planning
52 Graph changed by user
53 Graph changed by user
54 Towerskin02 populated
55 Graph changed by user
56 Fabrication Planning for Top
57 Graph changed by user
58 UBCTowerskin03populated
59 Floor shapes
60 Graph changed by user
61 fabrication Planningfloor-notworking

☐ Re-Fit
User changes: ☑ Monitor ☐ Have occurred
TransactionFile Console
Element Selection > Identify element to add to set

4.31

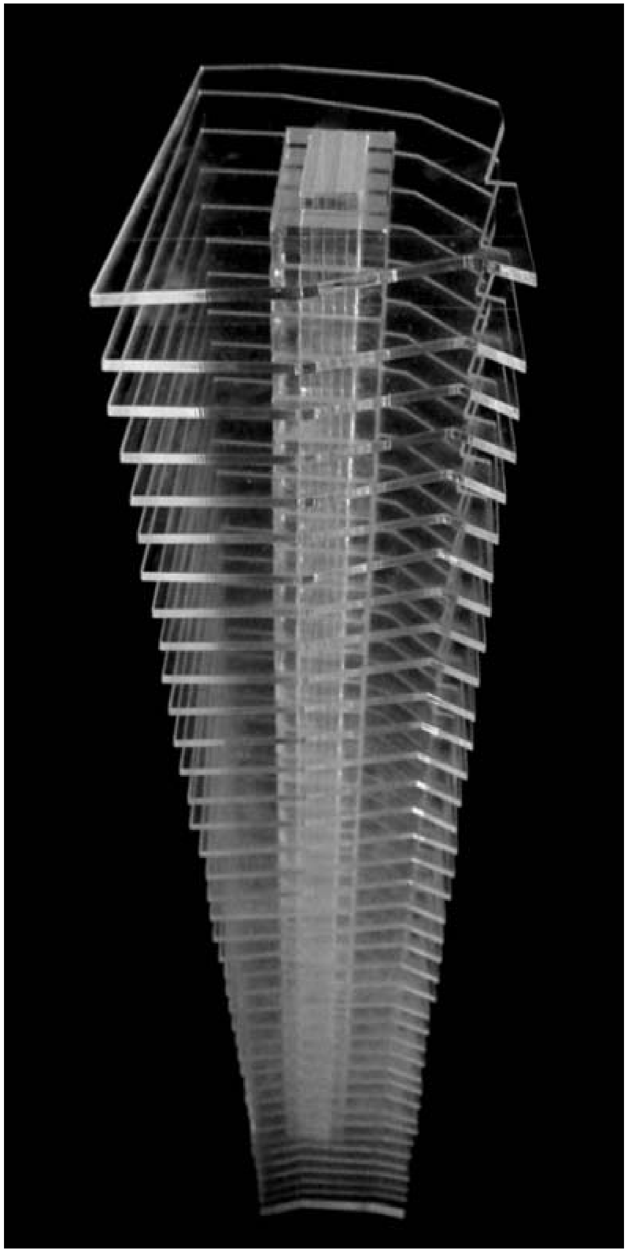

View 1 - Symbolic - Top
6 - GenerativeComponents Beta
File Features Toggle Graph Tools Debug Help
baseCS.XYplane
Default

Transaction
19 Tower face 2 cut linked to buildings
20 Graph changed by user
21 Tower position adjusted
22 Generatefeature type GC.UBC_BuildingResponse
23 Graph changed by user
24 Graph changed by user
25 Tower face 3 cut linked to buuldings
26 Tower face 3 cut adjusted
27 Graph changed by user
28 Graph changed by user
29 Graph changed by user
30 Graph changed by user
31 Graph changed by user
32 Graph changed by user
33 Compass created
34 Sun response done
35 Graph changed by user
36 Graph changed by user
37 Graph changed by user
38 Graph changed by user
39 LawCurve points created on edge2
40 Graph changed by user
41 Graph changed by user
42 Shape populated on face 2 Lawcurve
43 Shape populated on all faces
44 Graph changed by user
45 Graph changed by user
46 Consolidation of 2transactions
47 Graph changed by user
48 Park surface
49 Graph changed by user
50 Graph changed by user
51 Fabrication Planning
52 Graph changed by user
53 Graph changed by user
54 Towerskin02 populated
55 Graph changed by user
56 Fabrication Planning for Top
57 Graph changed by user
58 UBCTowerskin03populated
59 Floor shapes
60 Graph changed by user
61 fabrication Planningfloor-notworking

☐ Re-Fit
User changes: ☑ Monitor ☑ Have occurred
TransactionFile Console
Display complete

4.32

Floor slabs `shape04`, `shape05`, `shape06`, `shape07` are created in preparation for its fabrication planning and laser cutting.



4.33 Physical Model of Tower Fromation

2.10 Comments:

The high-rise design model shows a positive integration of procedural thinking and computation into the process. Several design concerns were identified, and design decisions were taken to set the way the building should respond to each of the addressed concerns. The design decisions are then abstracted into a set of rules to be executed by the software. In this exploration, the approach succeeded to construct a system that is dynamically responsive to the addressed issues simultaneously, and any changes that happen to its references. The system goes beyond representing how it is dynamically affected by each of the references; it calculates and represents how all the effects react to each other. It accounts for all the influencing factors with different weights and finds the best accumulation of design decisions that responds to all. This process reveals and solves additional layers of design complexity.

It is possible that this designed system be applied to address the same concerns in different sites. The outcome of the model would change to reflect how the forces act on its local site, and how the building's design should respond. The system is also capable of addressing issues like skin-scale shading and reflecting devices. It can be further utilised in the programming of such devices to be responsive to the solar movement and its location within the building facades.

As with the previous exploration in domestic living spaces, after-the fact evaluation and elimination of the non-working produced form is not needed for the designed responsive system. The rules, which were set for the system to execute, include the design limitations needed to control the system’s actions, and thus the final form. This guarantees the production of working space configurations each time the system is manipulated.

This exploration shows architecture formation as a pure result of a process of rational methodology. The building is designed through the strict execution of procedures specifically set to control how the building reacts to different influences. The formation of the building marks an objective research study on computation and its relation to the design process. However, subjective aspects of design were partially reserved for designers shown in their control over the articulation of setting the rules.

Other aspects, which were not included in the formation of this study, could contribute to a total change in the resulting form. Cultural issues presented in user preferences could also negotiate what is acceptable as a buildable design and what is not. The resulting form of architecture would always be a negotiation of the meaningful and the aesthetics as conceived by the persons involved in the process.

The incorporation of constraints into the process of design informs the design process by directing the system's performance towards the designer’s intended direction. This approach also adds the unexpected aspects of digital form making, while being strategic and conscious about the formation. It is capable of producing responsive architecture that is capable of addressing a multitude of design concerns, and it is only limited by the designer's dedication to feed the system. A rather simple design decision cumulatively turns into an increasingly complex one once the system’s references multiply.

[8] Vancouver Views: Council Approved View Cones. 18 10 2008 <http://vancouver.ca/commsvcs/views/viewcones/91.htm>.

5 Shade

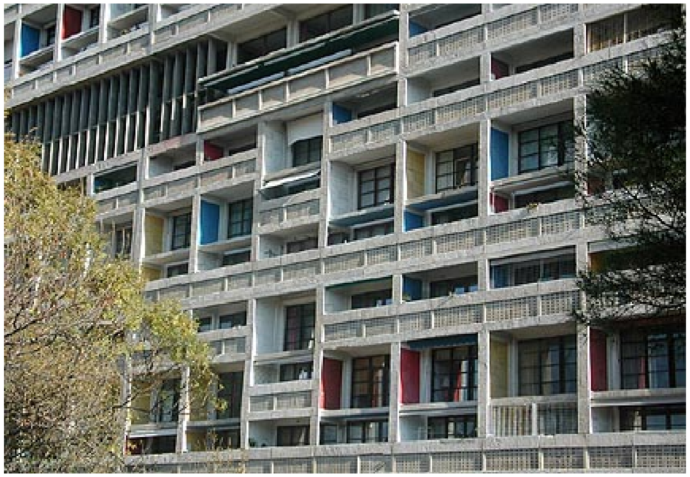
Introduction

This exploration focuses on buildings' skin. It is a proposal for a responsive sun shading system that uses parametric and computational techniques to change its form by strictly following an explicit programmatic function. Through a simple and direct approach, it also illustrates how rule based procedures can be utilised to add functional patterns to buildings' facades.

Design Approach

The rule based design approach, used in previous explorations, is used in this exploration to produce programmatic and formal complexity out of a simple initial state. First, the design element is abstracted into simple geometries that are modeled into the parametric system. Second, the design program is translated into a simple set of rules that guide the performance of geometries within the parametric system, and are stored inside the system's model. These rules dictate how the system responds to its changing influences. Third, changes in the outside influences trigger a dynamic response in the system, guaranteed by the strict execution of the explicitly set rules. At any instance of the model, form emerges as a result of the system's responsiveness to its changing influences. In this case, varying forms are always a representation of how the system strictly follows its designed function.

The system sets a relationship between the cordial directions (orientation) and the horizontal and vertical overhangs composing the shading system.



5.1 Unite d‘ Habitation, Marseilles, France. 1952

Design Background

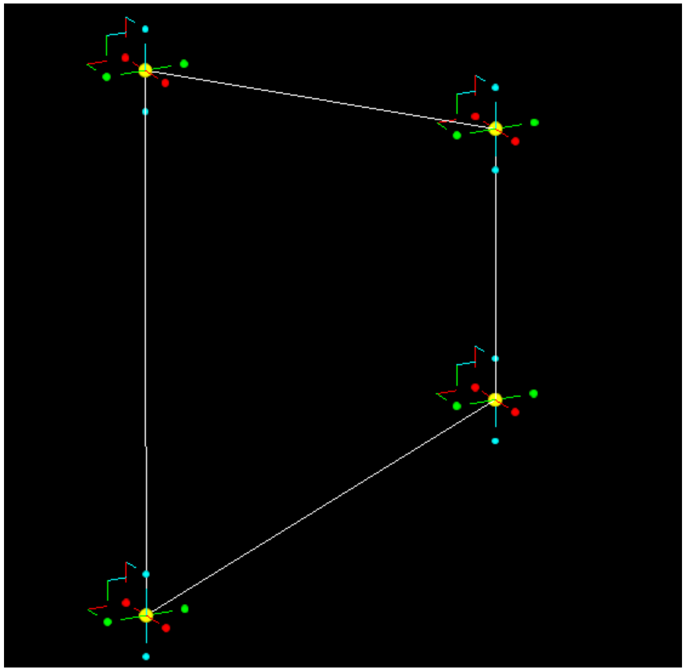
Inspired by Le Corbusier's modernist design of egg-crate like shading overhangs that he used in a number of his built projects, I decided to investigate how a parametric modeling can introduce a functional progression to shading overhangs. For instance, in Unite d‘ Habitation, Le Corbusier used a modular concrete screen wall "Brise Soleil" to reduce the solar heat gain entering the building through windows. The screen wall is composed of horizontal and vertical shading elements that have the same projection length. This egg-crate type of screen walls combine the characteristics of both horizontal and vertical shading elements and offer a considerable potential for solar control. On the other hand, the use of shading elements of the same depth questions the efficiency of such a system to provide the appropriate shading on each of the glazing surfaces.

Position

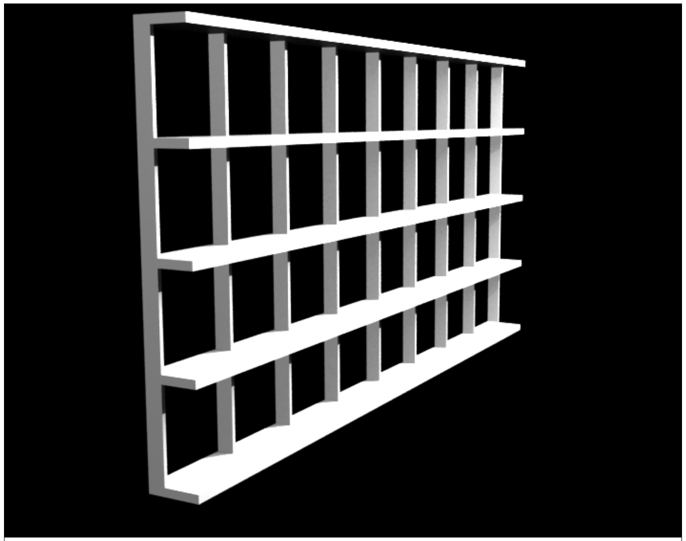
I based my proposal for the parametric system on the fact that vertical fins are generally assumed appropriate for providing solar control on East and West facades, while horizontal overhangs provide the best solar control on south facing facades[9]. The application of appropriate lengths of horizontal and vertical elements maximises the effect of egg-crate shading devices.

Objective

The shading system is designed to take into consideration the wall's orientation and accordingly decide the appropriate lengths of horizontal and vertical shading elements to be added to the wall as a screen system. The system should be able to respond dynamically to changes in its position relative to the North direction.



5.2 Base Shape



5.3 Shadying Sytem

Process

The system's design process starts with abstracting an exterior wall into its basic shape: a rectangle.

A series of overhangs are then added to the base rectangle. The maximum depth of the overhang (Lmax) is dependent on the number of subdivisions and therefore the subdivision length (S). Where:
Lmax = S

The actual length of the overhang (L) depends on its position in relation to the North direction. A factor (f), ranging from 0 to 1, reflects this relation and is multiplied to the max length to achieve the actual length. Where:
L = Lmax * f

Influencing Factors:
1. North direction

2. Thickness of overhangs (Constant)

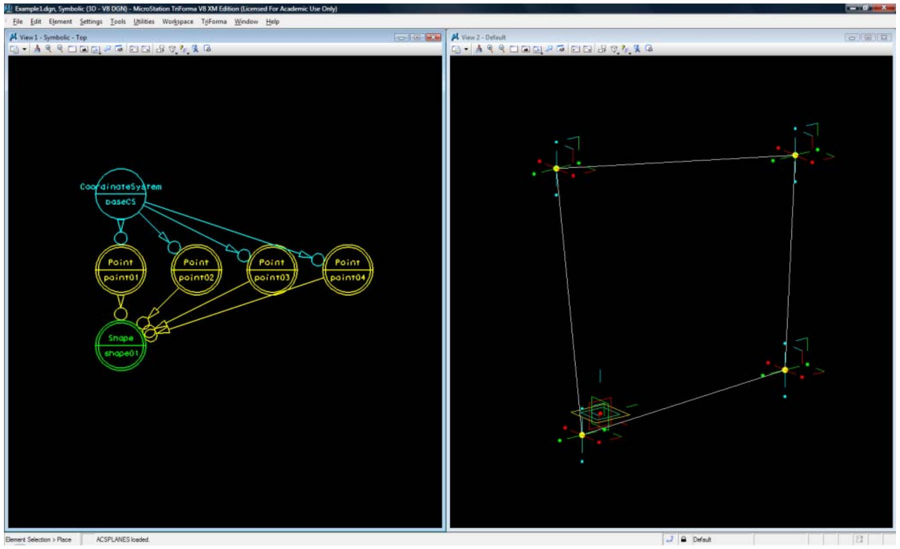
Rules:
1. North facing walls: No shading Overhangs (f=0)
2. South facing walls: The horizontal overhangs will have the max allowed length (f=1), and there will be no vertical overhangs (f= 0).
3. East + West facing walls: The vertical overhangs will have the max allowed length (f=1), and there will be no horizontal overhangs (f= 0_

System making

The following pages illustrate the process of building the system and integrating the design constraints using the parametric modeling software, *GenerativeComponents*.

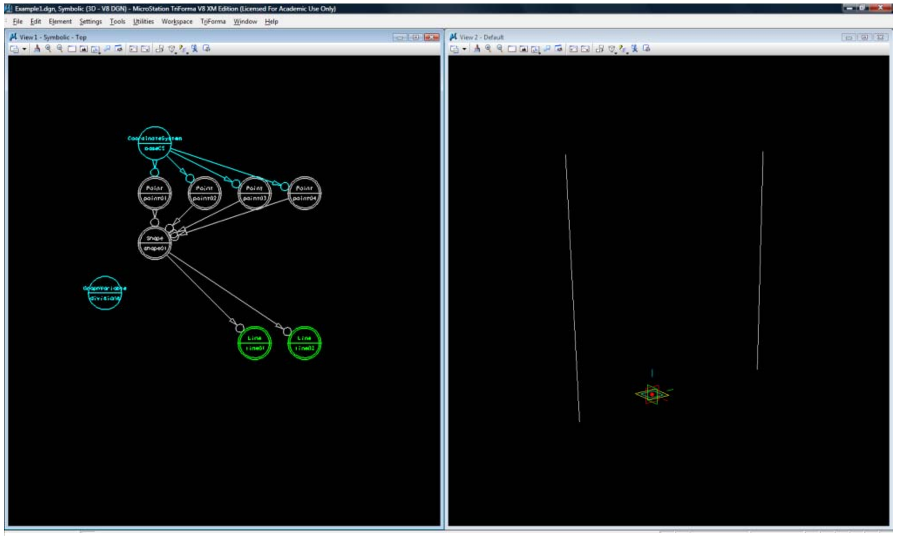
```
transaction modelBased "points and shape"
{
    feature point01 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (0.900256062153973);
        Ytranslation              = <free> (-1.17106949772331);
        Ztranslation              = <free> (-2.77555756156289E-17);
        Display                   = DisplayOption.Display;
        HandleDisplay             = DisplayOption.Display;
    }
    feature point02 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (0.647841114966038);
        Ytranslation              = <free> (-1.6709146375336);
        Ztranslation              = <free> (6.06229618060974);
        Display                   = DisplayOption.Display;
        HandleDisplay             = DisplayOption.Display;
    }
    feature point03 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (0.786013913746437);
        Ytranslation              = <free> (4.64199706144227);
        Ztranslation              = <free> (5.1023700547652);
        Display                   = DisplayOption.Display;
        HandleDisplay             = DisplayOption.Display;
    }
    feature point04 GC.Point
    {
        CoordinateSystem          = baseCS;
        Xtranslation              = <free> (0.576798427016679);
        Ytranslation              = <free> (4.52586903255925);
        Ztranslation              = <free> (0.212020244180056);
        Display                   = DisplayOption.Display;
        HandleDisplay             = DisplayOption.Display;
    }
    feature shape01 GC.Shape
    {
        Vertices                  =
{point01,point02,point03,point04};
    }
}
```

```
transaction modelBased "2 vt lines"
{
    feature divisions GC.GraphVariable
    {
        Value                     = 4.0;
        LimitValueToRange         = true;
        RangeMinimum              = 1.0;
        RangeMaximum              = 4.0;
        RangeStepSize             = 1.0;
        SymbolXY                  = {99, 103};
    }
    feature line01 GC.Line
    {
        StartPoint                = shape01.Vertices[2];
        EndPoint                  = shape01.Vertices[3];
        Display                   = DisplayOption.Display;
    }
    feature line02 GC.Line
    {
        StartPoint                = shape01.Vertices[0];
        EndPoint                  = shape01.Vertices[1];
        Display                   = DisplayOption.Display;
    }
}
```



5. 4

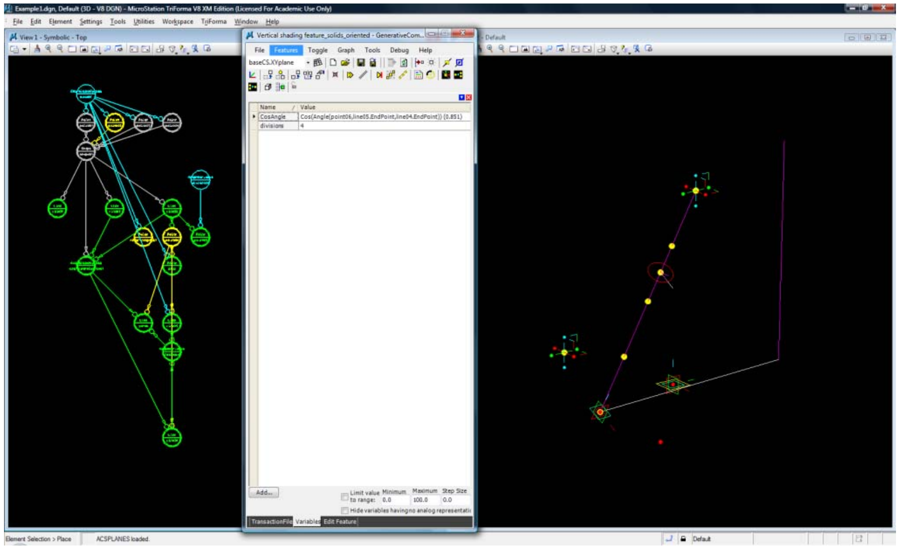
First, a rectangle is drawn on 4 points to be the base for the system



5. 5

Vertical lines are drawn on the sides of the shape to be a base for the subdivisions

```
transaction modelBased "lines from point"
{
       feature point05 GC.Point
    {
        Curve                          = line02;
        T                              = Series(0,1,1/divisions);
        SymbolXY                       = {104, 105};
        HandleDisplay                  = DisplayOption.Display;
    }
    feature point06 GC.Point
    {
        Curve                          = line02;
        T                              = <free> (0.630790283878103);
        SymbolXY                       = {103, 105};
        HandleDisplay                  = DisplayOption.Display;
    }
    feature coordinateSystem01 GC.CoordinateSystem
    {
        Origin                         = shape01.Vertices[0];
        PrimaryDirection               = line03;
        PrimaryAxis                    = AxisOption.Y;
        SecondaryDirection             = line02;
        SecondaryAxis                  = AxisOption.Z;
    }
    feature line03 GC.Line
    {
        StartPoint                     = shape01.Vertices[0];
        EndPoint                       = shape01.Vertices[3];
        SymbolXY                       = {100, 104};
    }
    feature line04 GC.Line
    {
        StartPoint                     = point06;
        Direction                      = coordinateSystem01.Xdirection;
        Length                         = .5;
        Color                          = 5;
    }
    feature line05 GC.Line
    {
        StartPoint                     = point06;
        Direction                      = baseCS.Xdirection;
        Length                         = .5;
        SymbolXY                       = {103, 108};
        SymbolicModelDisplay           = null;
        Color                          = 5;
        FillColor                      = -1;
        Free                           = true;
        Level                          = 0;
        LevelName                      = "Default";
        LineStyle                      = 0;
        LineStyleName                  = "0";
        LineWeight                     = 0;
        MaximumReplication             = true;
        RoleInExampleGraph             = null;
        Transparency                   = 1;
    }
    feature CosAngle GC.GraphVariable
    {
        Value                          =
Cos(Angle(point06,line05.EndPoint,line04.EndPoint));
        SymbolXY                       = {103, 109};
    }
    feature LineLength GC.Line
    {
        StartPoint                     = point06;
        Direction                      = coordinateSystem01.Xdirection;
        Length                         = CosAngle;
        SymbolXY                       = {102, 108};
        SymbolicModelDisplay           = null;
        Color                          = 0;
        FillColor                      = -1;
        Free                           = true;
        Level                          = 0;
        LevelName                      = "Default";
        LineStyle                      = 0;
        LineStyleName                  = "0";
        LineWeight                     = 0;
        MaximumReplication             = true;
        RoleInExampleGraph             = null;
        Transparency                   = 1;
    }


}
```



Name	Value
CosAngle	Cos(Angle(point06,line05.EndPoint,line04.EndPoint)) (0.851)
divisions	4

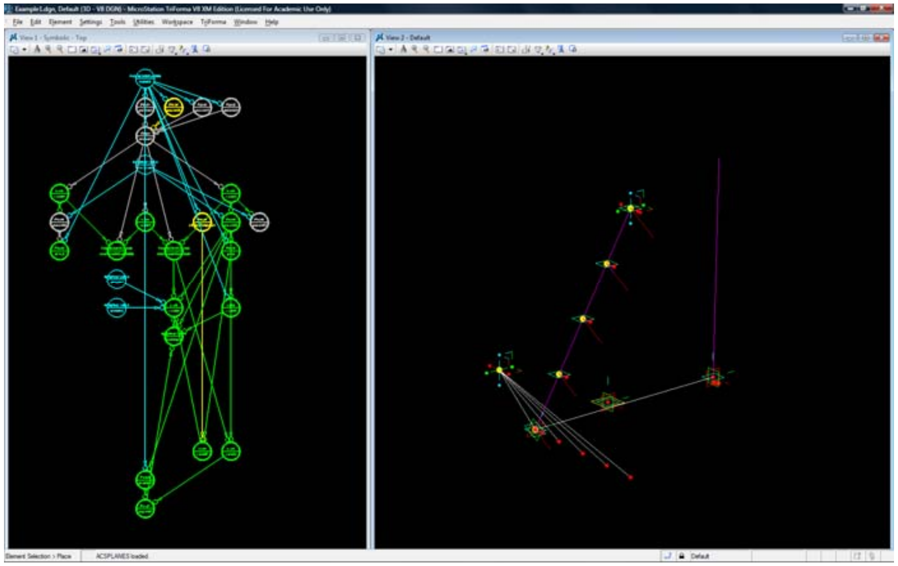
5. 6

Points (Point05) are drawn to mark the division on the vertical line. The spacing is controlled by *Variable* ***`divisions`***.

A coordinate system is set on the vertical line to determine its orientation relative to the North (X direction). Two directional lines (Line 04, Line05) start from a point on the vertical line, one has the default North direction and the other has a direction perpendicular to the vertical line.

Variable ***`CosAngle`*** is set to determine the Cosine of the angle between the Line04 and Line 05, which I will use as a factor to control the overhangs length.

```
transaction modelBased "projection line"
{
feature proj GC.Point
    {
        Plane                       = baseCS.XYplane;
        PointToProjectOnToPlane     = point06;
        SymbolXY                    = {103, 106};
    }
    feature referencepoint GC.Point
    {
        CoordinateSystem            = baseCS;
        Xtranslation                = <free> (-5.42881925419483);
        Ytranslation                = <free> (-1.00912222452869);
        Ztranslation                = <free> (0.0);
        SymbolXY                    = {102, 105};
        HandleDisplay               = DisplayOption.Display;
    }
}

    feature line07 GC.Line
    {
        StartPoint                  = referencepoint;
        EndPoint                    = proj;
    }
}



transaction modelBased "shadeV2"
{
    feature LineLength GC.Line
    {
        Length                      = CosAngle+line07.Length*shadeV2;
    }
    feature shadeV2 GC.GraphVariable
    {
        Value                       = 0.08;
        LimitValueToRange           = true;
        RangeMaximum                = 1.0;
        RangeStepSize               = 0.0;
        SymbolXY                    = {99, 108};
    }
}

transaction modelBased "Graph changed by user"
{
    feature point08 GC.Point
    {
        Curve                       = line01;
        T                           = Series(0,1,1/divisions);
        SymbolXY                    = {99, 105};
        HandleDisplay               = DisplayOption.Display;
    }
}

transaction modelBased "Prepare to generate feature type
GC.UBC_LinebyCosAngleLength08"
{
    feature CosAngle GC.GraphVariable
    {
        Value                       =
Cos(Angle(point06,line05.EndPoint,point07));
    }
    feature plane01 GC.Plane
    {
        CoordinateSystem            = baseCS;
        Xtranslation                = point06.X;
        Ytranslation                = point06.Y;
        Ztranslation                = point06.Z;
    }
    feature point07 GC.Point
    {
        Plane                       = plane01;
        PointToProjectOnToPlane     = line04.EndPoint;
    }
}
```

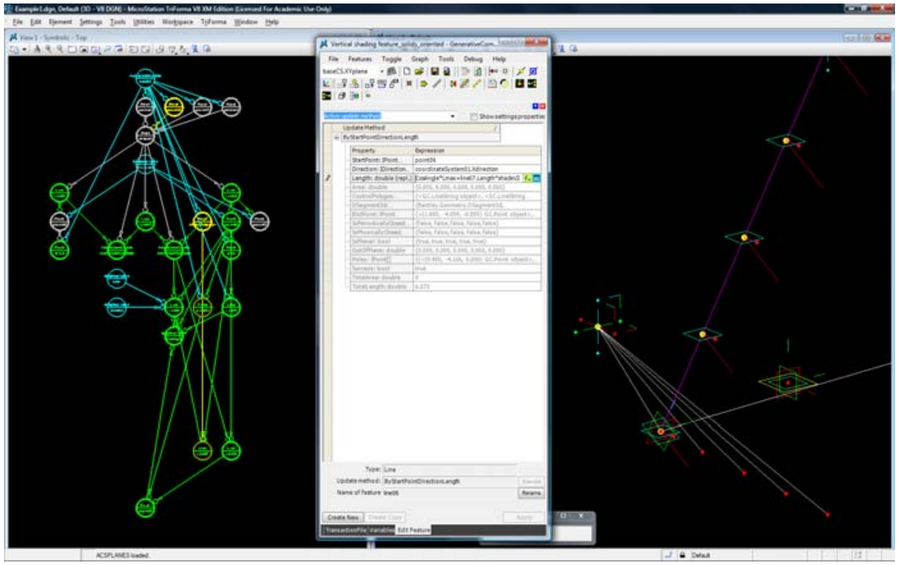


5.7

To determine the planer angle between the directional lines (Line 04, Line05), a projection of Line 04 is constructed to the horizontal plane. Variable Cos Angle is adjusted to measure Cosine the angle between two directional lines that lie on the same horizontal plane.

The division points (point06) are projected on the default horizontal plane, and linked to reference point lying on the same plane. The distance between the projection points and the reference point (line07.Length) is used to vary the overhangs' lengths. When the sideline is vertical, the overhangs would have the same lengths. When the sideline is slightly inclined, the overhangs would have varying lengths because of their varying orientations.

```
transaction modelBased "Prepare to generate feature type
GC.UBC_LinebyCosProjectedAngleLength09"
{
    deleteFeature LineLength;
    feature Lmax GC.GraphVariable
    {
        Value                     = .5;
        LimitValueToRange         = true;
        RangeMaximum              = 3.0;
        RangeStepSize             = 0.0;
        SymbolXY                  = {99, 107};
    }
    feature L GC.Line
    {
        StartPoint                = point06;
        Direction                 = coordinateSystem01.Xdirection;
        Length                    =
CosAngle*Lmax+line07.Length*shadeV2;
        SymbolXY                  = {101, 108};
    }

    feature coordinateSystem02 GC.CoordinateSystem
    {
        Origin                    = shape01.Vertices[3];
        PrimaryDirection          = line03;
        PrimaryAxis               = AxisOption.Y;
        SecondaryDirection        = line01;
        SecondaryAxis             = AxisOption.Z;
        SymbolXY                  = {99, 106};
    }
    feature point06 GC.Point
    {
        T                         = Series(0,1,1/divisions);
    }
    feature proj2 GC.Point
    {
        Plane                     = baseCS.XYplane;
        PointToProjectOnToPlane   = point08;
        SymbolXY                  = {97, 106};
    }
}
```

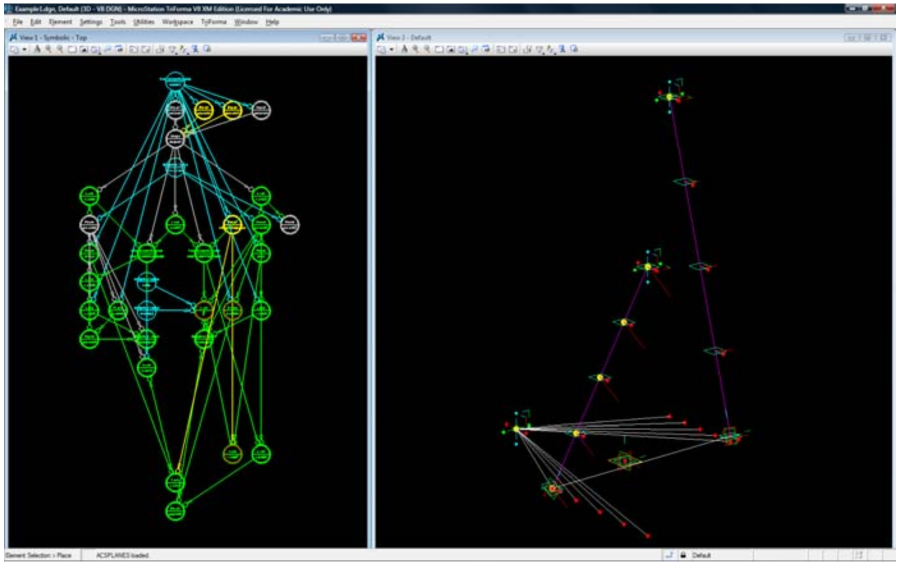


Property	Expression
StartPoint: IPoint...	point06
Direction: IDirection...	coordinateSystem01.Xdirection
Length: double (repl.)	CosAngle*Lmax+line07.Length*shadeV2

5.8

The overhangs' length:

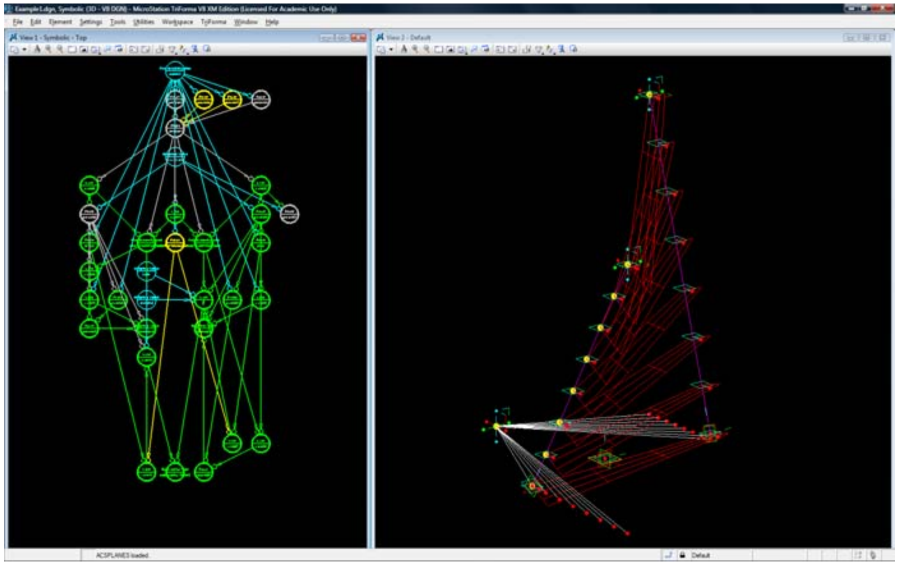
L = Lmax * Cosine Angle + Projection line Length (line07.Length) * 0.1


Example1.dgn, Default (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
View 2 - Default
Element Selection > Place
ACSPLANES loaded
Default

5.9

Divisions and overhang directional lengths are constructed on the other sideline using the same sequence.

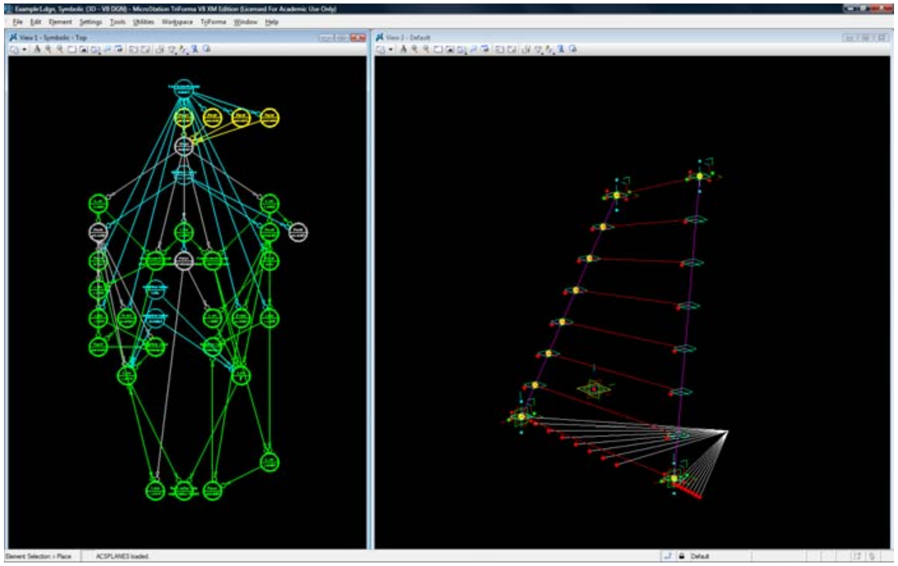
```
transaction modelBased "surfaces between lines -south wall works"
{
    feature L GC.Line
    {
        Length                    =
Abs(CosAngle*Lmax+line07.Length*shadeV2);
    }

    feature bsplineSurface05 GC.BsplineSurface
    {
        StartCurve                = line10;
        EndCurve                  = L;
        Color                     = 3;
    }
    feature divisions GC.GraphVariable
    {
        Value                     = 7.0;
        RangeMaximum              = 10.0;
    }
}
```


Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
View 2 - Default
ACSPLANES loaded.
Default

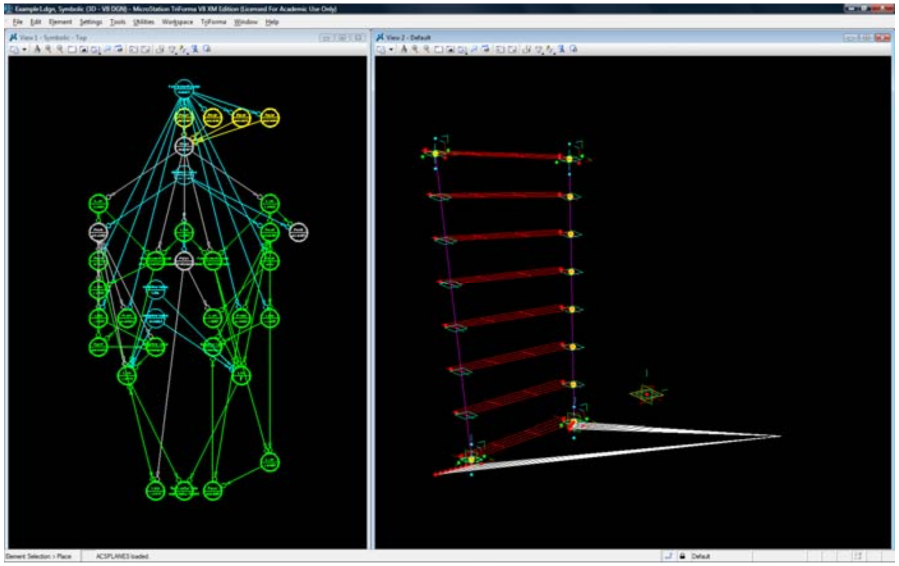
5.10

Surfaces are constructed between the overhang length lines on the sidelines.

```
transaction modelBased "East facing wall test"
{
    feature baseCS GC.CoordinateSystem
    {
        Display                    = DisplayOption.Display;
    }
    feature line10 GC.Line
    {
        Length                     =
Abs(cosAngle2*Lmax+line11.Length*shadeV2);
    }
    feature point03 GC.Point
    {
        Ytranslation               = <free> (-4.34766465394409);
    }
    feature point04 GC.Point
    {
        Ytranslation               = <free> (-4.17338362707951);
    }
    feature shadeV2 GC.GraphVariable
    {
        Value                      = 0.0;
    }
}

transaction modelBased "South facing wall test"
{
    feature point02 GC.Point
    {
        Xtranslation               = <free> (1.36936699571921);
        Ytranslation               = <free> (-4.67742589723736);
    }
    feature point03 GC.Point
    {
        Xtranslation               = <free> (2.55293201974526);
        Ytranslation               = <free> (-11.8859752799482);
    }
    feature point04 GC.Point
    {
        Xtranslation               = <free> (2.48515215470223);
        Ytranslation               = <free> (-10.1328428456147);
    }
}


transaction modelBased "length equation changed"
{
    feature L GC.Line
    {
        Length                     =
Abs(CosAngle)*Lmax+line07.Length*shadeV2*Abs(CosAngle);
    }

    feature louverthickness GC.GraphVariable
    {
        Value                      = .15;
    }
    feature solid01 GC.Solid
    {
        SurfaceToOffset            = bsplineSurface05;
        OffsetAboveSurface         = 0;
        OffsetBelowSurface         = louverthickness;
        Color                      = 3;
    }
}

transaction modelBased "solids"
{
    feature solid01 GC.Solid
    {
        OffsetAboveSurface         = louverthickness;
        OffsetBelowSurface         = 0;
    }

}
```



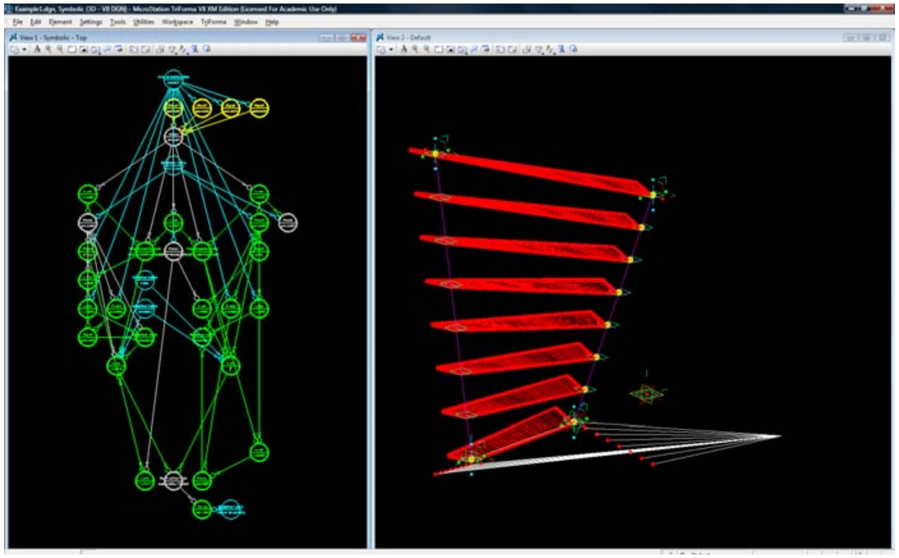
5.11



5.12

Changes to the wall orientation are done to test how the horizontal overhang system responds to East (Figure 5.11) and South (Figure 5.12).

```
transaction generateFeatureType "Generate feature type GC.UBC_VerticalShading07"
{
    type                      = GC.UBC_VerticalShading07;
    loadIntoFutureSessions    = true;
    inputProperties           = {
                                    property GC.IPoint baseCS
                                    {
                                        feature                  = baseCS;
                                        isReplicatable           = true;
                                        isParentModel            = true;
                                    }
                                    property GC.Shape shape01
                                    {
                                        feature                  = shape01;
                                    }
                                    property GC.IPoint referencepoint
                                    {
                                        feature                  = referencepoint;
                                    }
                                    property double louverthickness
                                    {
                                        feature                  = louverthickness;
                                        isReplicatable           = true;
                                    }
                                    property double divisions
                                    {
                                        feature                  = divisions;
                                        isReplicatable           = true;
                                    }
                                    property double Lmax
                                    {
                                        feature                  = Lmax;
                                    }

                                };
    outputProperties          = {


                                    property GC.Solid solid01
                                    {
                                        feature                  = solid01;
                                        isDynamic                = true;
                                    }
                                };
    internalProperties        = {
                                    property GC.BsplineSurface bsplineSurface05
                                    {
                                        feature                  =

bsplineSurface05;

                                        isConstruction           = true;
                                        isDynamic                = true;
                                    }

                                    property GC.GraphVariable CosAngle
                                    {
                                        feature                  = CosAngle;
                                        isConstruction           = true;
                                        isDynamic                = true;
                                    }
                                    property GC.GraphVariable cosAngle2
                                    {
                                        feature                  = CosAngle2;
                                        isConstruction           = true;
                                        isDynamic                = true;
                                    }

                                };
}
```



5.13

Solids are constructed on the overhang surfaces. The thickness of the overhang is manipulated by *Variable* **`thickness`**.

A shading feature (GC.UBC_VerticalShading07) is generated and stored inside the program's default features list. This feature stores the process of designing the horizontal shading system, including the relationships and calculations between the features (elements) that build the process.

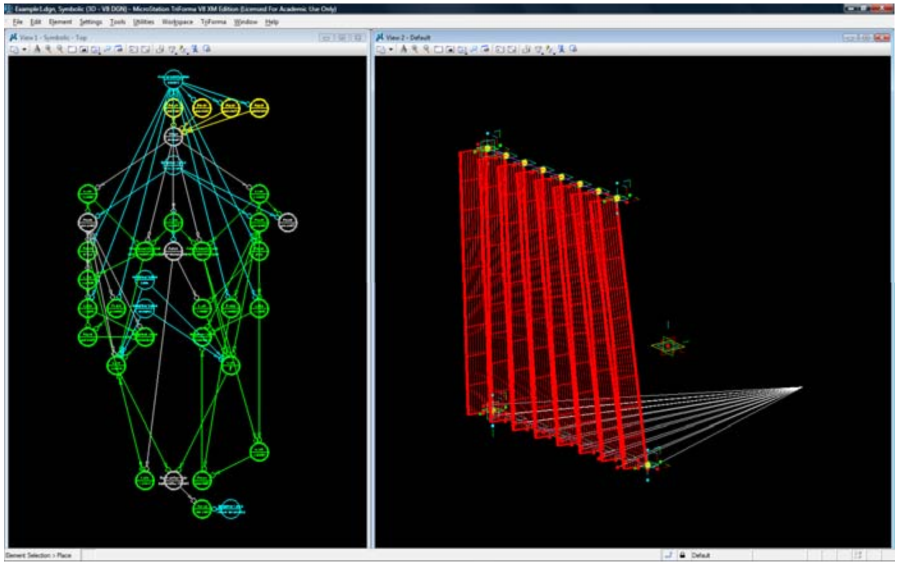
The feature generation enables the shading system's model to be duplicated on any shape in future, after specifying the base shape and the input variables.

Inputs:

1. Base Coordinate system
2. Base shape
3. Reference point
4. Louver thickness
5. Number of divisions
6. Maximum Louver Length

Outputs:

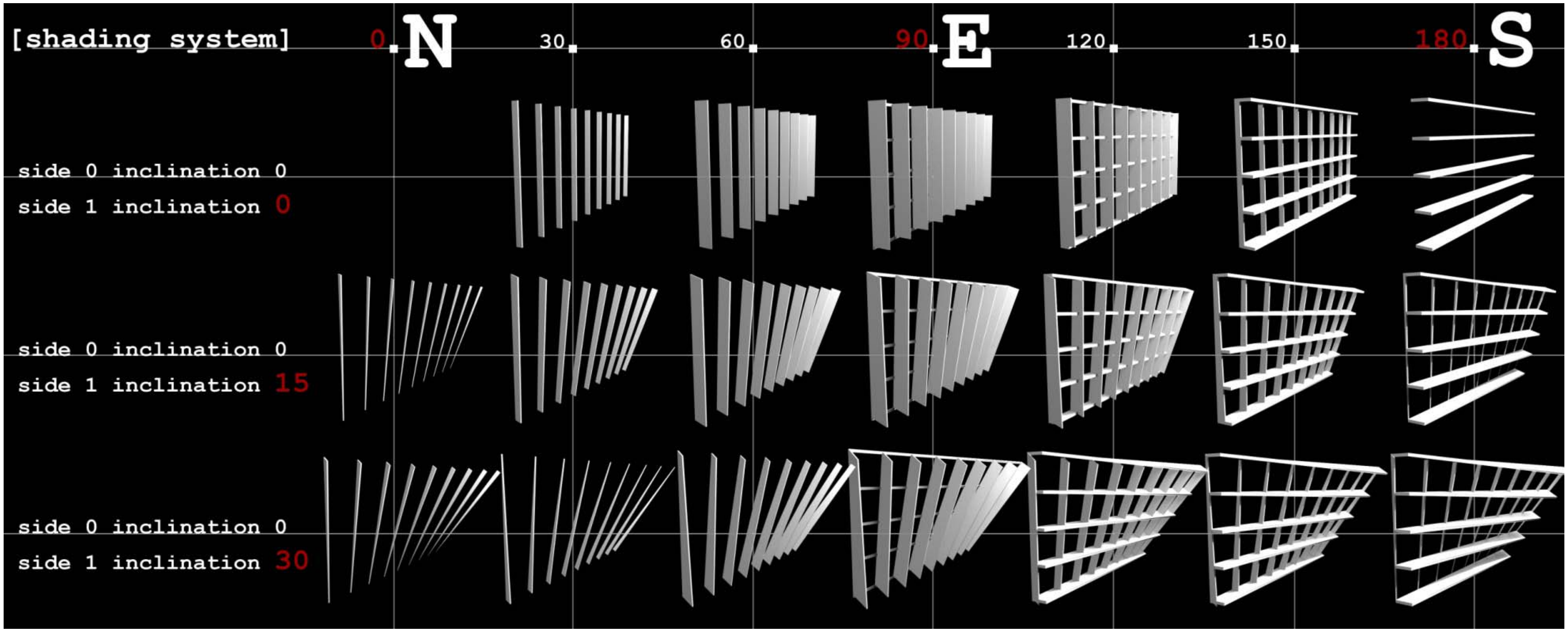
1. Louver's solids


Example1.dgn, Symbolic (3D - V8 DGN) - MicroStation TriForma V8 XM Edition (Licensed For Academic Use Only)
File Edit Element Settings Tools Utilities Workspace TriForma Window Help
View 1 - Symbolic - Top
View 2 - Default
Element Selection > Place
Default

5.14

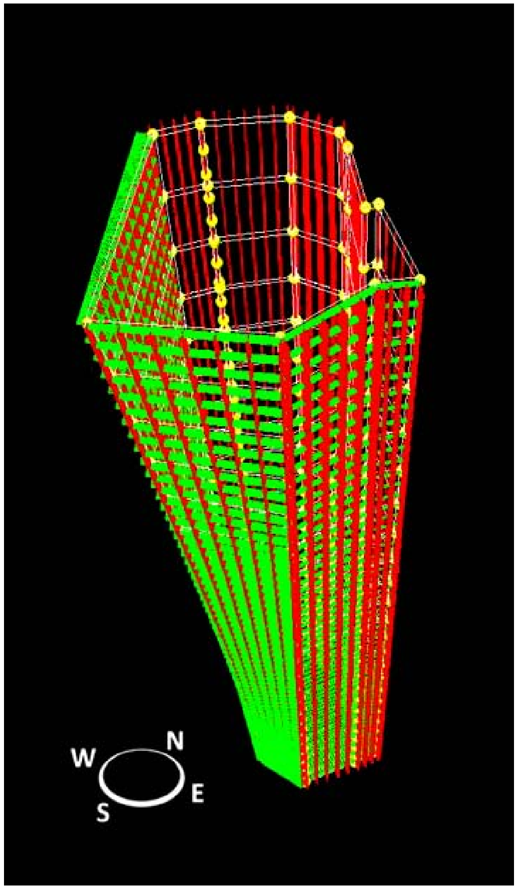
Following a similar process, a system for the vertical shades is generated as a feature in the program. This system controls the way vertical overhangs vary their lengths according to the orientation relative to the north direction.

A combination of both the horizontal and vertical features creates the proposed parametric egg-crate shading system. The power of this system is most appreciated in how it calculates the states in between the main cordial directions and applies the best fitting lengths of overhangs according to the orientation of the wall.



5.15

Figure 5.15 shows the complete shading system applied to a screen wall. Orientation angles are mapped out on 30 degrees increments. The diagram shows in the first row how the parametric shading system responds to the different orientations by gradually changing its state from one configuration to the other. A North-facing wall with no shading overhangs gradually changes to a vertical configuration facing East, and then to a horizontal configuration facing South. In the second and third rows, one side of the wall inclines so the wall takes a three dimensional shape. The system works in this case to adjust each of the overhangs' start and end lengths for a better shading configuration.



5.16

Application

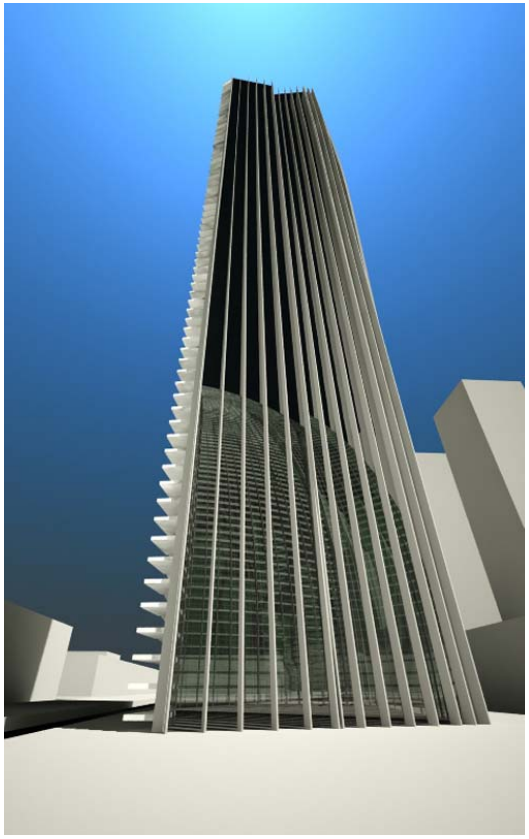
The shading system is applied to the “Tower Formation” exploration of the previous chapter. A shape polygon is populated on the exterior of the tower’s structure to serve as a base for applying the sun shading system. The populated shape polygon consists of a series of four sided polygons that vary in dimensions and angles to form a second skin based on the tower’s forming points. Each of the shape polygons is one floor in height.

The different orientations on each of the shapes that form the tower’s folded structure hold a potential to test the application of the shading system. The system automatically detects the orientation of each of the tower faces, and accordingly calculates the relevant combination of horizontal and vertical overhangs on each face.

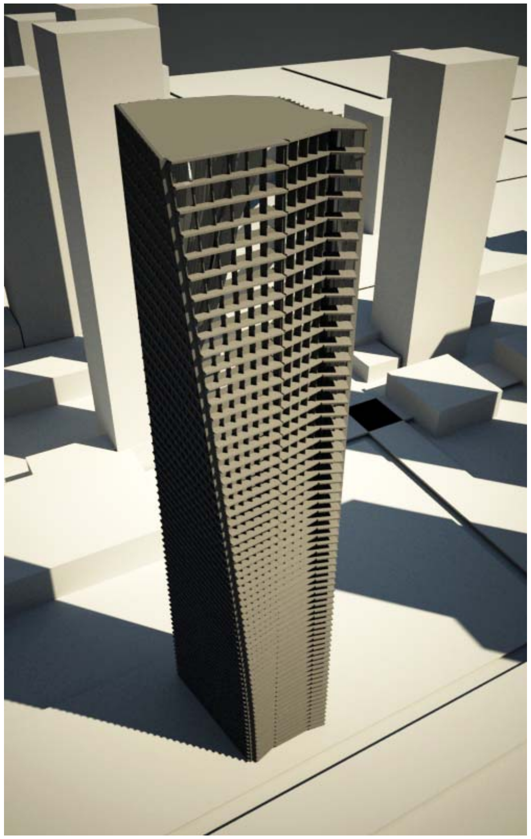
The vertical divisions in the system are set to four, so there are three horizontal overhangs on the facade between one floor and the other. Vertical spacing between the overhangs is equal to the horizontal spacing.

Figure 5.16 shows the system’s population over the tower’s exterior polygons. The vertical overhangs (fins) are shown in red, while the horizontal ones are shown in green.

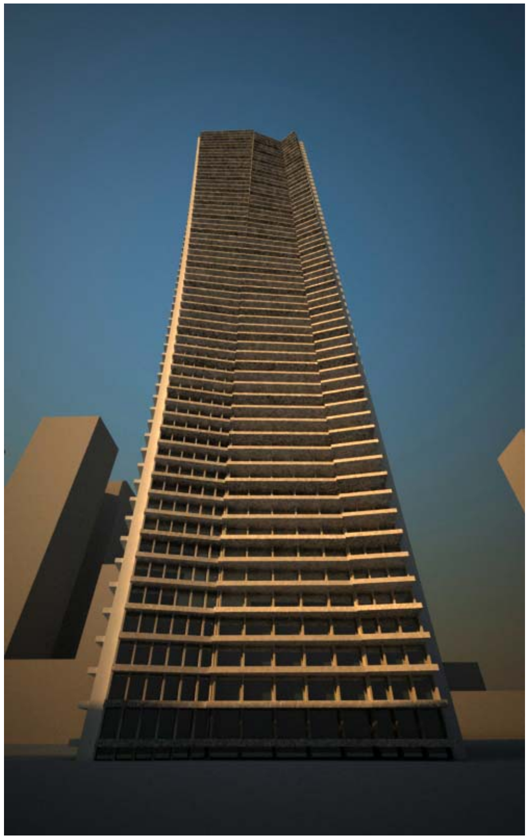
The illustration clearly shows the system response to its application on different faces of the tower. South-facing sides trigger the system to add the horizontal shading elements with varying lengths. Since the entire faces of the tower are facing West or East directions with different angles, the system adds vertical fins on all the faces with varying overhang lengths. The renders in figures 5.17 to 5.21 give a better understanding of the systems application over the tower’s structure.



5.17



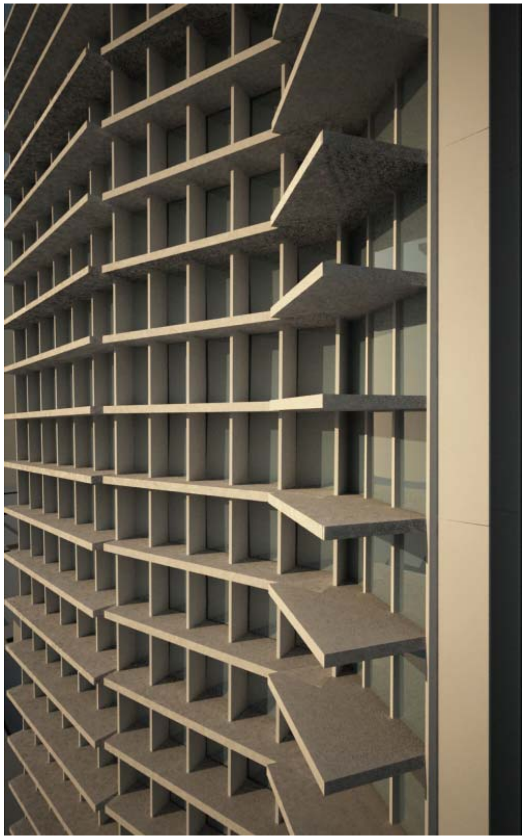
5.18



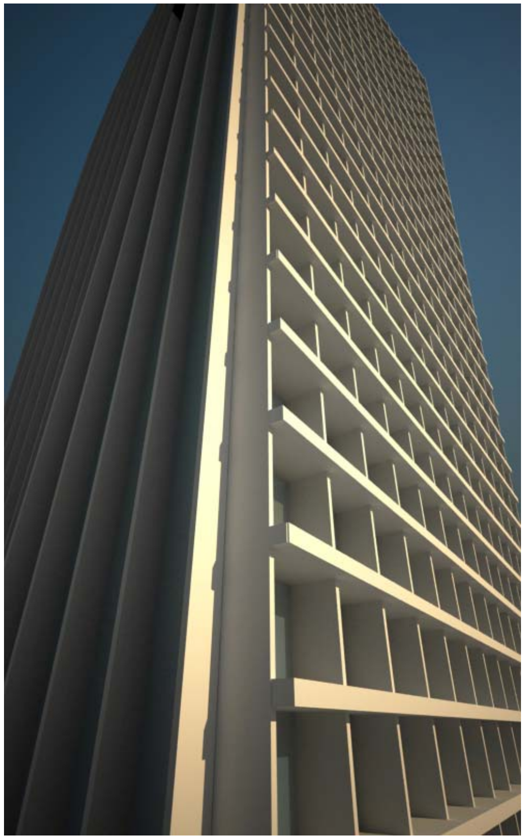
5.19

Figure 5.17 show the North East side of the tower after the application of the shading system screen. This side shows a lack of horizontal overhangs and a number of vertical ones due to its orientation: North-facing facades do not require the horizontal shading elements, thus the lack of them.

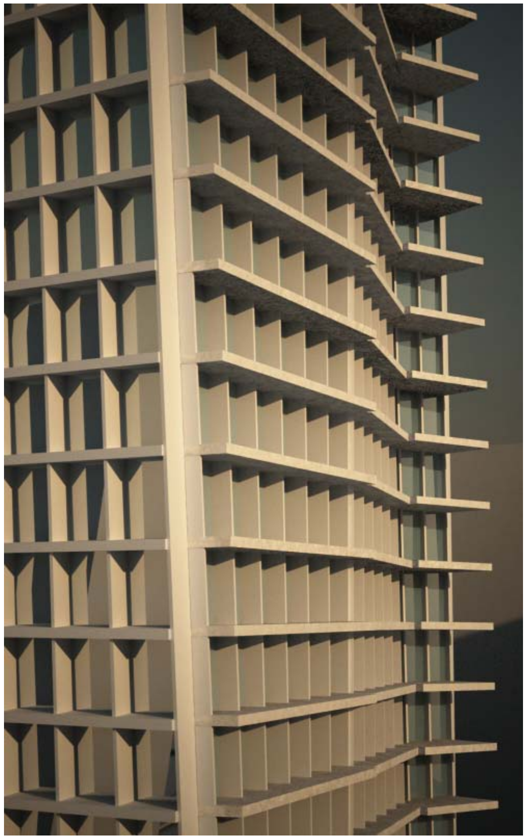
Similarly, on the South East side of the tower shown in figures 5.18 and 5.19, the system provides both horizontal and vertical overhangs to the face of the tower to accommodate for both the South and East directions.



5.20



5.21



5.22

The above renders show the varying lengths of overhangs in better detail. Figure 5.20 shows a portion of a generally southeast- facing facade with folds that slightly change the orientation angel of the faces. It is obvious how the systems calculates and thus decides the varying lengths of vertical and horizontal overhangs in this view. The lengths gradually change as the faces change from an orientation that mostly faces south to one that is more directed towards the southeast. In a similar way, the shading system reacts to both the northwest and southwest orientations in figure 5.21. It accordingly decides the best fitting configuration for both faces. Figure 5.22 shows both the Southwest and Southeast sides of the tower. It is clear that the Southwest side shown on the left of the image is directly facing southwest because both the horizontal and vertical overhangs have the same lengths in this instance. On the right side of the image in figure 5.22, the tower's folds illustrate the different reactions of the shading system to the folded faces.

Conclusion

The work in this chapter shows the application of a rule based design approach on the formation of sun shading system. The functional significance of the system is stated at the beginning of the exploration: The system should use the orientation of screen walls to adjust horizontal and vertical louvers in the way they provide the best sun control. Rules guiding how the system performs its proposed function were explicitly stated at the beginning of the exploration. The system is modeled using parametric modeling software through integrating the rules and their constraints into the design process. Parametric modeling allows for the system's dynamic response to its changing design drivers. Changes done to the screen wall's orientation initiates new calculations in the shading system. These calculations are visually translated into new lengths of the overhangs that compose the shading system.

The designer has control on the starting form of the elements composing the system, the criteria to which the system responds, and on how the system responds to these criteria. However, the parametric system decides and generates the final form of the shading device.

The rule based design approach used in this system has the potential to create a real-time responsive shading system that detects the sun's position and accordingly calculate the appropriate overhang lengths. By using the right combination of light sensors, materials, and mechanical systems, this new shading system can be applied to building's facades to optimise the use of shading devices and maximise solar control while allowing for appropriate exposure.

[7] Cole, Ray. "Environmental Systems and Control I."

6 City Configurations

Introduction

“City Configurations” define a responsive approach to city planning and explores how parametric modeling can help establish a dynamic way of planning the relationships between buildings in the city. The exploration makes use of parametric modeling to model the existing and future relationships between buildings. It uses these relationships to define the buildings’ form and location at an urban scale. The output of this exploration can be used as an application for city planners to map out the existing buildings in a particular neighbourhood and set the planning rules guiding the regulations of both existing and new buildings. Accordingly, the proposed parametric application generates a dynamic and responsive regulation for new buildings.

Design Approach

Zoning bylaws are a set of rules created by city planners in order to regulate current and future developments in the city. The rule-based design approach used in this thesis holds great potential at the urban planning scale, because direct and explicit laws generally guide the urban planning process. Parametric modeling is useful for executing multiple procedures and controlling multiple processes, which opens the possibility to develop more complex bylaw based on dynamic relationships between buildings.

The main idea of this exploration is to integrate the planning laws into a parametric system to create a responsive model of current and future buildings in the city. This system would follow the planning rules to regulate and calculate forms, heights, and locations of future buildings. This approach would encourage planners to set laws that not only regulate a building according to its static site location, but also taking into account its dynamic relationship with the surrounding buildings.

As in the previous explorations, the rule based design approach is used to produce programmatic and formal complexity out of simple initial states. First, neighbourhood buildings are abstracted into simple geometries that are modeled into the parametric system. Second, regulating planning laws are proposed and translated into a set of rules that guide the performance of geometries within the parametric system. These rules are stored inside the system's model and dictate how the system responds to its changing influences. Third, changes in the system's references trigger a dynamic response in the system, guaranteed by the strict execution of the explicitly set rules. At any instance of the model, configurations emerge because of the system's responsiveness to its changing influences.

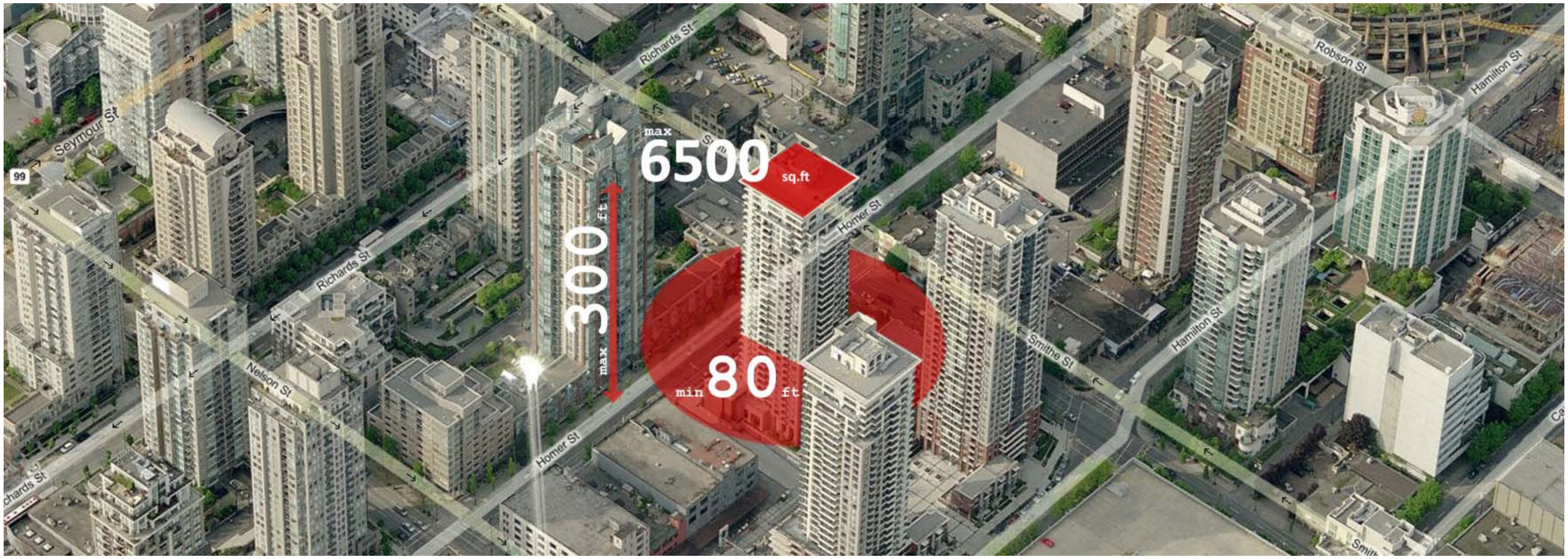
Design Background

The City of Vancouver regulates the design of new developments in the city through their Land Use and Development Policies and Guidelines. The city divides Vancouver's downtown into several districts or neighbourhoods, and each has its own regulations. The site for this exploration is the larger context of the tower's exploration in chapter four; Vancouver's Downtown South.

The city regulations and design guidelines[10] generally emphasise on:

1. Public and private views: Views to the north mountains are greatly appreciated and reserved in Vancouver. Guidelines regarding the preservation of these views from public spaces regulate the maximum heights of buildings as dictated by preserved view corridors throughout the downtown area. The maximum allowable height in Downtown South is 300 ft.

The views from private residential units to the mountains are preserved by the general orientation of the downtown street grid. The grid is rotated by a 45 degree angle to the north in order to generate maximum north view exposure. A maximum floor plate area of 6500 sq.ft. is set constraint the size of the buildings, in order to minimise blocking of views by surrounding towers. Compact slim towers are generally encouraged. (Figure 6.1)



6.1 City of Vancouver guidelines and regulations.

2. Light and ventilation: The Downtown South design guidelines emphasises on minimising shadow impact on parks, public open spaces, and major streets. This reflects on the required separation of 80 ft. between towers in the city core. Larger separations are encouraged to let natural lighting penetrate between the buildings to open gathering spaces.

3. Continuous street edge definition: The City of Vancouver calls for a consistent pattern in the street edge definition to produce a lively residential community inside the city core. Their objective to create safe, active and attractive streets with visual interest for pedestrians is promoted by assigning townhouses podiums to each residential tower.

4. Privacy: Providing appropriate privacy for buildings is accomplished by regulating the amount of setbacks needed for each development site.

Position

I based the rules for my planning parametric system on the shortcomings of current city planning regimes in the City of Vancouver Downtown South. I believe each building exists in a network of neighbouring buildings and that the relationship between them should start to affect how new projects are reviewed.

With deploying the rule based design approach to this particular project I critique:

1. Fixed planning regulations: The bylaws, regulations, and design guideline that govern the configuration of existing and new buildings are based on fixed rules that are applied to all buildings in neighbourhood with fixed boundaries. These rules are not designed to consider the relationship between buildings nor the effect they have on each other.

2. Individual project assessment: During the developing permit stage, each new building project is assessed individually according to how it follows the regulations and guidelines. I believe there should be a dynamic regulating relationship between buildings in the city. In a way, direct neighbouring buildings should have more effect on particular building projects than buildings in its larger neighbourhood context.

Proposal

A parametric planning program can help the formation of city configurations, by establishing a network of regulations that controls the relationship between city towers in a neighbourhood. The program's objective is to integrate a wide range of planning rules and regulations that constrains the relationship between towers. The system should be able to change the proposed projects' configurations according to the dynamic state of their surrounding context.

Process

The process of creating this parametric system started by the investigation of critical planning and design criteria that could positively regulate the relationship between city towers. Based on the shortcomings found in the City of Vancouver's Downtown South design guidelines, the proposed parametric system promotes a method of how these shortcomings can be addressed.

Criteria:

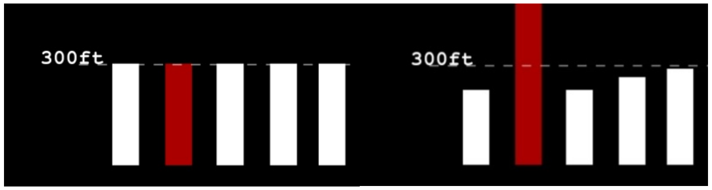
The proposed system regulates the configuration of buildings based on the following criteria:

1. Fixed density:

The city planning guideline set fixed maximum allowable heights and floor plate areas to control the desired density in the city core. These regulations are applied to all buildings inside a particular district, regardless of the current and future situation of its immediate context. Instead of applying fixed heights and areas, my approach is to set regulations for a desired fixed density and for districts or neighbourhood inside the city core. We could achieve a constant density inside one district by limiting the collective volume of buildings in that particular area.

$\sum V = \sum (H \times A) = \text{constant}$

If we assume floor plate areas are fixed, then the collective building volume is directly proportional to individual buildings' heights. The idea is to control the height relationship between towers in a district to reflect the maximum allowable volume and density. The maximum allowable height is distributed on all buildings instead of regulating the heights per project. This allows for an account of context and surroundings to occur between the city buildings: a dynamic relationship is set to regulate heights so if one existing tower is at a more than average height, we know the system would change the configuration of surrounding new towers to reflect the needed decrease in height.



6.2 Fixed Density

Rule:

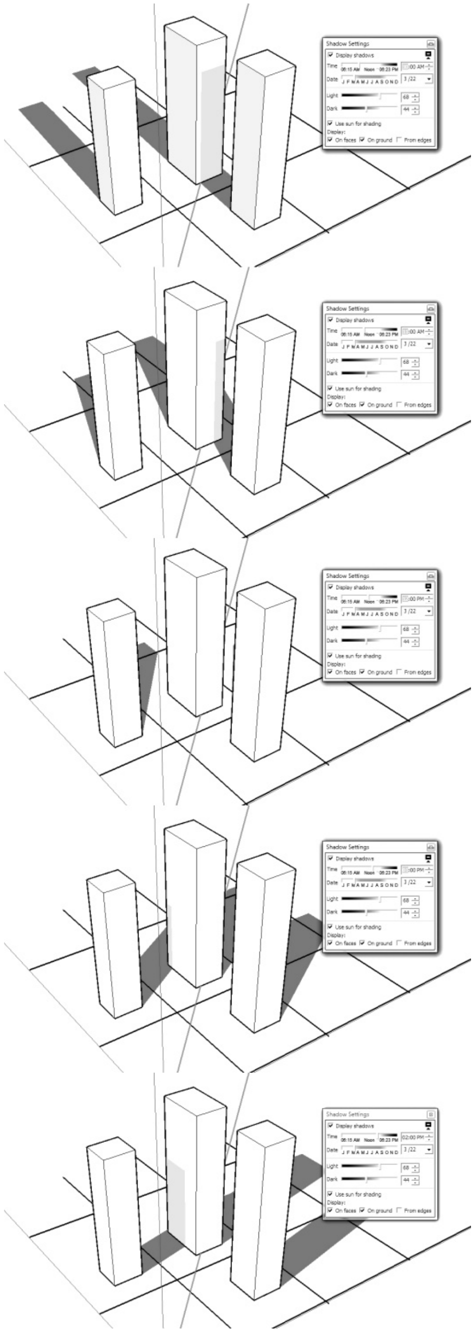
$\sum H = H1 + H2 + H3 + H4{+}{+} = \text{constant}$

This collective approach to regulating heights applies a non-linear change to the maximum allowable heights. In other words, one particular building's height affects its immediate neighbours the most by changing their allowable heights. The effect gradually fades the further the distance is. (Figure 6.2)

2. Shading Impact:

The Downtown South design guidelines lack specific rules to regulate the relationship between buildings regarding natural lighting and the shading impact. A shadow analysis is currently required by the city to review the shading impact of a proposed building on its surroundings, in order to minimise shadows on public open spaces. Developments over 35 feet in height require a shadow impact analysis taken at the equinox at 10:00 a.m., noon, 2:00 p.m., and 4:00 p.m.

Instead of reviewing the shading analysis for every building in an independent process, the proposed planning system models the shadow analysis together with rules that could regulate its effect on surrounding buildings. An existing building's shadow analysis could dictate the allowable positioning of future surrounding buildings in order to minimise shadow casting on building faces. In my proposed system, a dynamic relationship is modeled between buildings' heights, shadows, and locations. This dynamic relationship controls buildings' heights and positions to avoid shadow casting on building faces.



6.3 Shading study

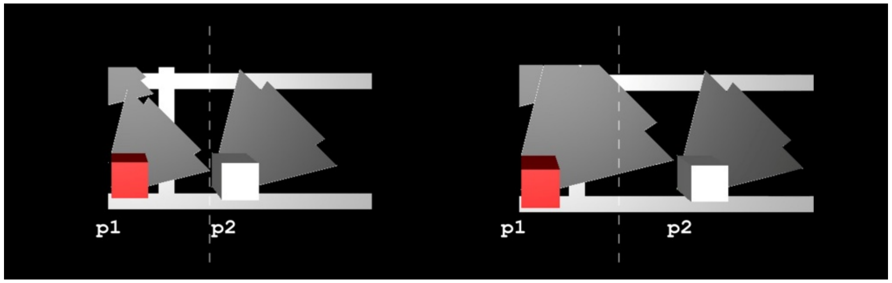
Rule:

No shadow casting on buildings should happen during 10:00 am to 2:00 pm between the March and September equinoxes.

A shadow analysis was performed on a building model to decide the footprint of shadows in relation to heights during the above mentioned times. (Figure 6.3) Because the building's shadow is proportional to its height and size, the shadow footprint is used in the parametric system to control the required and allowable heights and separations between buildings. In the dynamic parametric model, a building is positioned to avoid shadow casting on neighbouring buildings. (Figure 6.4)

p2 = p1 + L shading

This dynamic model responds to changes in heights, position, and footprint of any of the buildings, and accordingly set the allowable configurations of other buildings in the neighbourhood.



6.4 Shading Impact

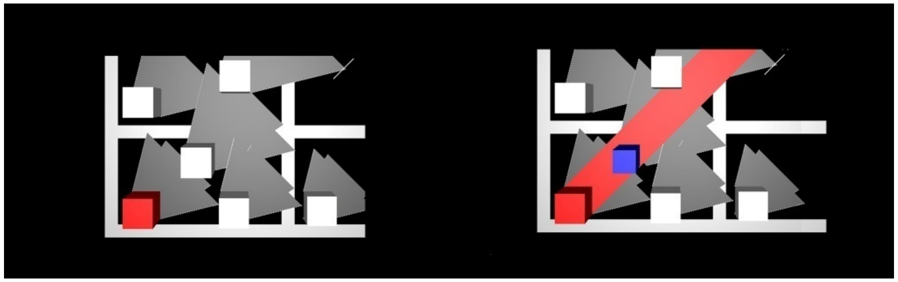
3. View Blocking:

As the view to the north mountains is strongly encouraged in the City of Vancouver's design guidelines, the proposed system controls the relationship between buildings to avoid view blocking to the north. The system calculates the building's orientation towards north and scans the surrounding buildings. If a proposed building's position is in the way of other buildings' views, the system will automatically reconfigure the allowable floor plate size of the future building to minimise the blocking.

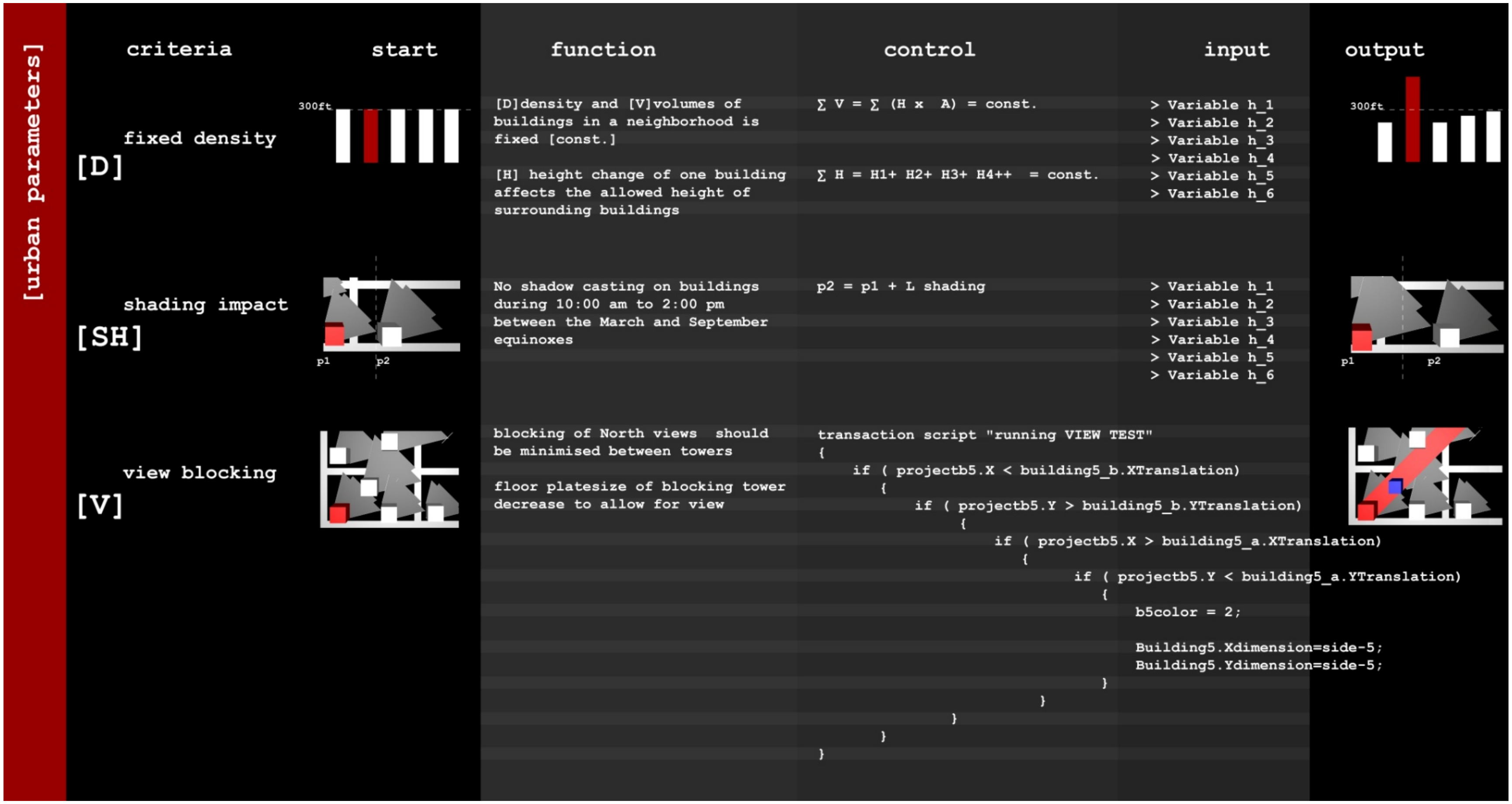
Rules:

1. Blocking of North views should be minimised between towers.
2. Floor plate size of blocking tower should decrease to allow for view.

A view test is done in the proposed system by extending lines from one building's face towards the north direction. If the extension lines hit other buildings' locations then the view is obstructed. In this case, the allowable floor plate size for the obstructing buildings is decreased. (Figure 6.5)



6.5 View Blocking Test



criteria | start | function | control | input | output

fixed density

[D]

[D]density and [V]volumes of buildings in a neighborhood is fixed [const.]

$\sum V = \sum (H \times A) = const.$

[H] height change of one building affects the allowed height of surrounding buildings

$\sum H = H1+ H2+ H3+ H4++ = const.$

> Variable h_1
> Variable h_2
> Variable h_3
> Variable h_4
> Variable h_5
> Variable h_6

shading impact

[SH]

No shadow casting on buildings during 10:00 am to 2:00 pm between the March and September equinoxes

p2 = p1 + L shading

> Variable h_1
> Variable h_2
> Variable h_3
> Variable h_4
> Variable h_5
> Variable h_6

view blocking

[V]

blocking of North views should be minimised between towers

floor platesize of blocking tower decrease to allow for view

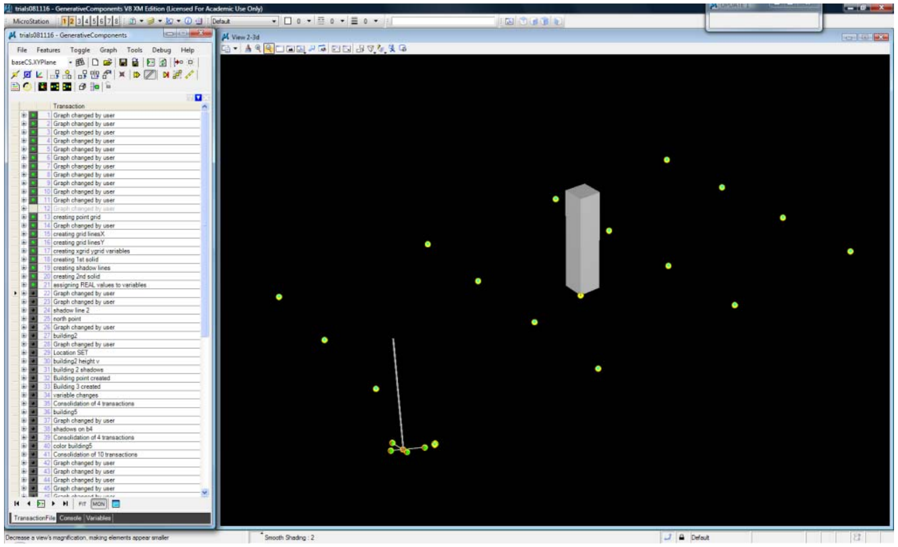
```
transaction script "running VIEW TEST"
{
    if ( projectb5.X < building5_b.XTranslation)
        {
            if ( projectb5.Y > building5_b.YTranslation)
                {
                    if ( projectb5.X > building5_a.XTranslation)
                        {
                            if ( projectb5.Y < building5_a.YTranslation)
                                {
                                    b5color = 2;

                                    Building5.Xdimension=side-5;
                                    Building5.Ydimension=side-5;
                                }
                        }
                }
        }
}
```

6.6 City Configurations Parameters

```
transaction modelBased "creating point grid"
{
    feature point07 GC.Point
    {
        CoordinateSystem          = baseCS;
        XTranslation              = Series(0,2*xblock,xblock);
        YTranslation              = Series(0,3*yblock,yblock);
        ZTranslation              = <free> (0.0);
        SymbolXY                  = {104, 100};
        Color                     = 0;
        FillColor                 = -1;
        Free                      = true;
        HandlesVisible            = true;
        IsModifiable              = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        Replication               =
ReplicationOption.AllCombinations;
        SymbologyAndLevelUsage    =
SymbologyAndLevelUsageOption.AssignToFeature;
        Transparency              = 1;
    }
    feature xblock GC.GraphVariable
    {
        Value                     = 25;
        RangeStepSize             = 0.0;
        SymbolXY                  = {105, 101};
    }
    feature yblock GC.GraphVariable
    {
        Value                     = 10;
        RangeStepSize             = 0.0;
        SymbolXY                  = {105, 100};
    }
}

transaction modelBased "creating 1st solid"
{
    feature csBuilding1 GC.CoordinateSystem
    {
        CoordinateSystem          = baseCS;
        XTranslation              = point07[1][1].XTranslation+10;
        YTranslation              = point07[1][1].YTranslation+2.5;
        ZTranslation              = 0;
        SymbolXY                  = {100, 110};
        Color                     = 0;
        FillColor                 = -1;
        Free                      = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        SymbologyAndLevelUsage    =
SymbologyAndLevelUsageOption.AssignToFeature;
        Transparency              = 1;
    }
    feature h1 GC.GraphVariable
    {
        Value                     = 20;
    }
    feature solid01 GC.Solid
    {
        CoordinateSystemAtOrigin  = csBuilding1;
        Xdimension                = 3;
        Ydimension                = 3;
        Zdimension                = h1;
    }
    feature xgrid GC.GraphVariable
    {
        Value                     = 75;
    }
}
```

System making

The following illustrates the process of creating the 'City Configurations' parametric system in *GenerativeComponents*. First, through a process of reduction and abstraction, elements like streets and buildings inside a downtown district are modeled as basic geometries. The three design criteria regarding density, shading, and view are then integrated as rules and relationships that guide the interaction between the system's elements. Manipulation of any of the system's elements or changing the criteria trigger a configuration adjustment of the whole model by following the rules and constraints previously set into the model.



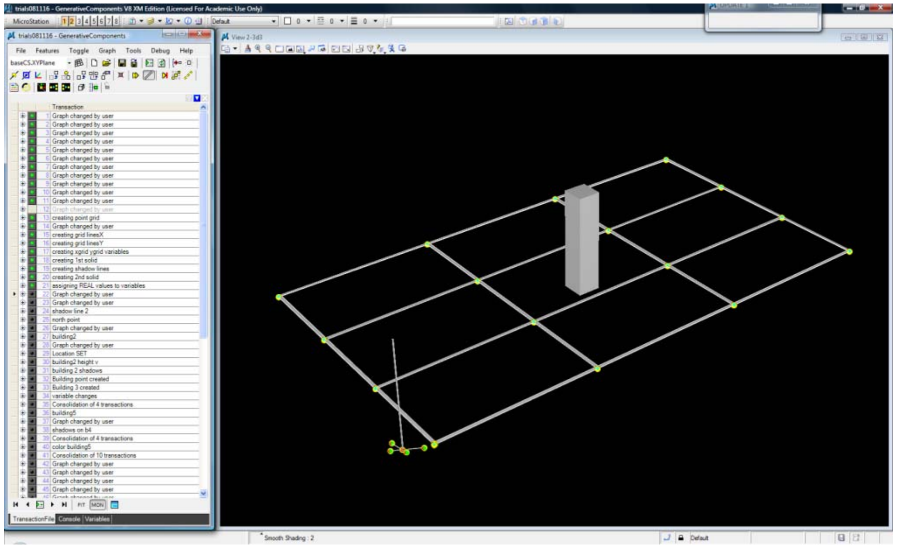
	Transaction
1	Graph changed by user
2	Graph changed by user
3	Graph changed by user
4	Graph changed by user
5	Graph changed by user
6	Graph changed by user
7	Graph changed by user
8	Graph changed by user
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	creating point grid
14	Graph changed by user
15	creating grid linesX
16	creating grid linesY
17	creating xgrid ygrid variables
18	creating 1st solid
19	creating shadow lines
20	creating 2nd solid
21	assigning REAL values to variables
22	Graph changed by user
23	Graph changed by user
24	shadow line 2
25	north point
26	Graph changed by user
27	building2
28	Graph changed by user
29	Location SET
30	building2 height v
31	building 2 shadows
32	Building point created
33	Building 3 created
34	variable changes
35	Consolidation of 4 transactions
36	building5
37	Graph changed by user
38	shadows on b4
39	Consolidation of 4 transactions
40	color building5
41	Consolidation of 10 transactions
42	Graph changed by user
43	Graph changed by user
44	Graph changed by user
45	Graph changed by user

Decrease a view's magnification, making elements appear smaller

6.7

A point grid (point07) represents street intersections. A solid is created to represent the first building in the neighbourhood.

```
transaction modelBased "creating xgrid ygrid variables"
{
    feature point07 GC.Point
    {
        XTranslation              = Series(0,xgrid,xblock);
        YTranslation              = Series(0,ygrid,yblock);
    }
    feature xblock GC.GraphVariable
    {
        LimitValueToRange         = true;
        RangeMaximum              = 50.0;
    }
    feature xgrid GC.GraphVariable
    {
        Value                     = 50;
        RangeStepSize             = 0.0;
    }
    feature yblock GC.GraphVariable
    {
        Value                     = 10.0;
        LimitValueToRange         = true;
        RangeMaximum              = 50.0;
    }
    feature ygrid GC.GraphVariable
    {
        Value                     = 30;
        RangeStepSize             = 0.0;
    }
}


transaction modelBased "Graph changed by user"
{
    feature road GC.GraphVariable
    {
        Value                     = 3;
        RangeStepSize             = 0.0;
    }
}

transaction script "creating grid linesX"
{
    {for (int i=0;i<5;i++)

                {
                Line line=new
Line("line_"+i).ByPoints(point07[0][i], point07 [5][i]);
                }
             {
            Line line=new Line("line__0").Offset(line_0, road,
baseCS.XZPlane );
              }

              {
            BSplineSurface BSplineSurface=new
BSplineSurface("surface__0").Ruled(line__0, line_0 );
                 }
                 {
            Line line=new Line("line__1").Offset(line_1, road,
baseCS.XZPlane );
              }
                   {
            BSplineSurface BSplineSurface=new
BSplineSurface("surface__1").Ruled(line__1, line_1 );
                 }
                 {
            Line line=new Line("line__2").Offset(line_2, road,
baseCS.XZPlane );
              }
                   {
            BSplineSurface BSplineSurface=new
BSplineSurface("surface__2").Ruled(line__2, line_2 );
                 }
                 {
            Line line=new Line("line__3").Offset(line_3, road,
baseCS.XZPlane );
              }
                   {
            BSplineSurface BSplineSurface=new
BSplineSurface("surface__3").Ruled(line__3, line_3 );
                 }
                 {
            Line line=new Line("line__4").Offset(line_4, road,
baseCS.XZPlane );
              }
                   {
            BSplineSurface BSplineSurface=new
BSplineSurface("surface__4").Ruled(line__4, line_4 );
                 }

            }
}
```



	Transaction
1	Graph changed by user
2	Graph changed by user
3	Graph changed by user
4	Graph changed by user
5	Graph changed by user
6	Graph changed by user
7	Graph changed by user
8	Graph changed by user
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	creating point grid
14	Graph changed by user
15	creating grid linesX
16	creating grid linesY
17	creating xgrid ygrid variables
18	creating 1st solid
19	creating shadow lines
20	creating 2nd solid
21	assigning REAL values to variables
22	Graph changed by user
23	Graph changed by user
24	shadow line 2
25	north point
26	Graph changed by user
27	building2
28	Graph changed by user
29	Location SET
30	building2 height v
31	building 2 shadows
32	Building point created
33	Building 3 created
34	variable changes
35	Consolidation of 4 transactions
36	building5
37	Graph changed by user
38	shadows on b4
39	Consolidation of 4 transactions
40	color building5
41	Consolidation of 10 transactions
42	Graph changed by user
43	Graph changed by user
44	Graph changed by user
45	Graph changed by user

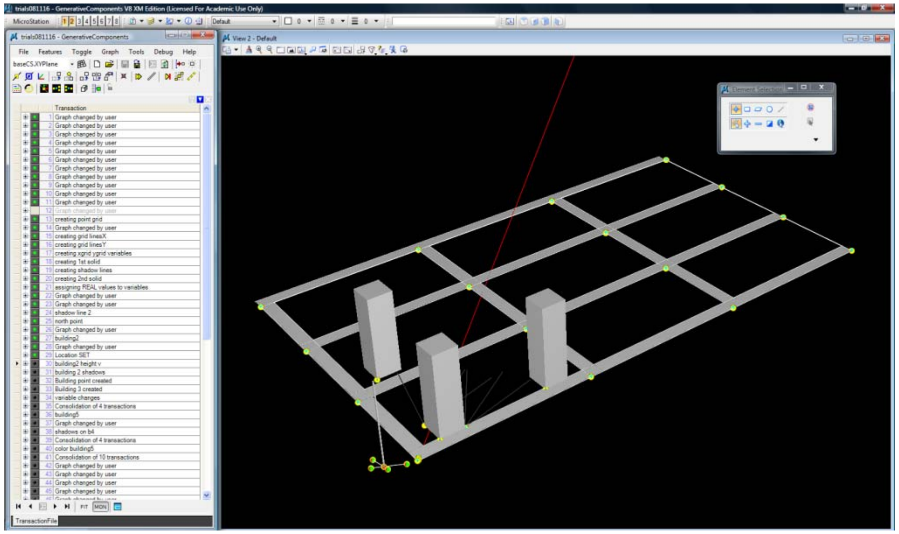
6.8

A series of lines pass through the initial grid point (point07) to define street edges. *Variables* ***Road*** controls the width of the streets.

The point grid spacing is adjusted to reflect the actual city block dimensions.

Variables ***Xblock, Xgrid, Yblock, Ygrid*** control the block size (increment) and the size of the area in study (end) both in X and Y directions.

```
transaction modelBased "creating shadow lines"
{
    feature building1shadowa GC.Line
    {
        StartPoint                = csBuilding1;
        Direction                 = direction01;
        Length                    = h1;
        SymbolXY                  = {101, 113};
        Color                     = 8;
        FillColor                 = -1;
        Free                      = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        Transparency              = 1;
    }
feature shadowdirection GC.Direction
    {
        Origin                    = point07[0][0];
        DirectionPoint            = point06;
        SymbolXY                  = {101, 112};
        SymbolicModelDisplay      = null;
        Color                     = 0;
        FillColor                 = -1;
        Free                      = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        PartFamilyName            = null;
        PartName                  = null;
        RoleInExampleGraph        = null;
        Transparency              = 1;
    }
transaction modelBased "creating 2nd solid"
{
    feature Building GC.Solid
    {
        CoordinateSystemAtOrigin  = csBuilding;
        Xdimension                = side;
        Ydimension                = side;
        Zdimension                = h1;
        SymbolXY                  = {100, 125};
        SymbolicModelDisplay      = null;
        Color                     = color;
        FillColor                 = -1;
        Free                      = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        PartFamilyName            = null;
        PartName                  = null;
        RoleInExampleGraph        = null;
        Transparency              = 1;
    }
    feature csBuilding GC.CoordinateSystem
    {
        CoordinateSystem          = baseCS;
        XTranslation              = point07[1][2].XTranslation+10;
        YTranslation              = point07[1][2].YTranslation+2.5;
        ZTranslation              = 0;
        ComputeGeometryInParameterSpace = null;
        SymbolXY                  = {100, 124};
        SymbolicModelDisplay      = null;
        Color                     = 0;
        FillColor                 = -1;
        Free                      = true;
        Level                     = 0;
        LevelName                 = "Default";
        LineStyle                 = 0;
        LineStyleName             = "0";
        LineWeight                = 0;
        MaximumReplication        = true;
        PartFamilyName            = null;
        PartName                  = null;
        RoleInExampleGraph        = null;
        SymbologyAndLevelUsage    =
SymbologyAndLevelUsageOption.AssignToFeature;
        Transparency              = 1;
    }
}
```



trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
Default
trials081116 - GenerativeComponents
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
View 2 - Default
FIT
MON
TransactionFile

	Transaction
1	Graph changed by user
2	Graph changed by user
3	Graph changed by user
4	Graph changed by user
5	Graph changed by user
6	Graph changed by user
7	Graph changed by user
8	Graph changed by user
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	creating point grid
14	Graph changed by user
15	creating grid linesX
16	creating grid linesY
17	creating xgrid ygrid variables
18	creating 1st solid
19	creating shadow lines
20	creating 2nd solid
21	assigning REAL values to variables
22	Graph changed by user
23	Graph changed by user
24	shadow line 2
25	north point
26	Graph changed by user
27	building2
28	Graph changed by user
29	Location SET
30	building2 height v
31	building 2 shadows
32	Building point created
33	Building 3 created
34	variable changes
35	Consolidation of 4 transactions
36	building5
37	Graph changed by user
38	shadows on b4
39	Consolidation of 4 transactions
40	color building5
41	Consolidation of 10 transactions
42	Graph changed by user
43	Graph changed by user
44	Graph changed by user
45	Graph changed by user

6.9

Buildings 2 and 3 are created and placed in the street grid according to their shading impact. Footprints of shadow casting are modeled for each building.

Building2.Ytranslation = Building1.Ytranslation

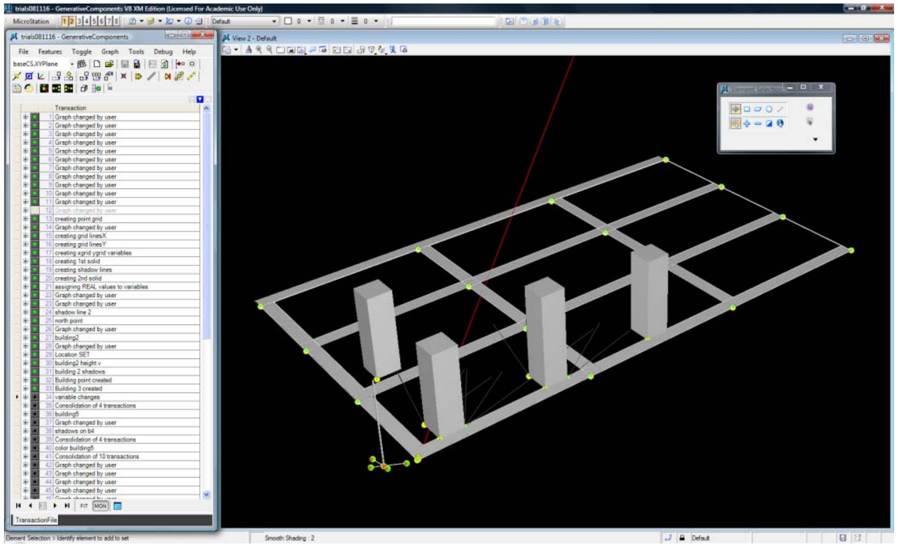
Building2.Xtranslation = Building1.Xtranslation + Building1. Side + SL1

Building3.Xtranslation = Building1.Xtranslation

Building3.Ytranslation = Building1.Ytranslation + Building1. Side + SL1

Where Building1.Side is the side dimension of building 1, and SL1 is the length building 1 shadow cast.

```
    transaction modelBased "variable changes"
{
    feature h1 GC.GraphVariable
    {
        Value                      = 70.0;
    }
    feature h2 GC.GraphVariable
    {
        Value                      = 60.0;
    }
    feature h3 GC.GraphVariable
    {
        Value                      = 55.75;
    }
feature building3_X GC.GraphVariable
    {
        LimitValueToRange          = true;
        RangeMinimum               = 20.0;
        SymbolXY                   = {86, 105};
    }
    feature building4_X GC.GraphVariable
    {
        Value                      = Abs(h1-(point07[0][1].Y-
point10.Y));
        LimitValueToRange          = true;
        RangeMinimum               = 20.0;
        RangeStepSize              = 0.0;
    }
    feature csBuilding GC.CoordinateSystem
    {
        YTranslation               = point07[0][1].Y+building4_X;
        SymbolXY                   = {91, 103};
    }
    feature csBuilding1 GC.CoordinateSystem
    {
        YTranslation               =
point07[0][0].YTranslation+road+ 10;
        SymbolXY                   = {93, 103};
    }
    feature csBuilding2 GC.CoordinateSystem
    {
        XTranslation               = point07[1][0].X-building2_X;
        SymbolXY                   = {89, 103};
    }
    feature csBuilding3 GC.CoordinateSystem
    {
        SymbolXY                   = {85, 103};
    }
}
```


trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
trials081116 - GenerativeComponents
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
View 2 - Default
FIT
MON
TransactionFile
Element Selection > Identify element to add to set
Smooth Shading : 2
Default

	Transaction
1	Graph changed by user
2	Graph changed by user
3	Graph changed by user
4	Graph changed by user
5	Graph changed by user
6	Graph changed by user
7	Graph changed by user
8	Graph changed by user
9	Graph changed by user
10	Graph changed by user
11	Graph changed by user
12	Graph changed by user
13	creating point grid
14	Graph changed by user
15	creating grid linesX
16	creating grid linesY
17	creating xgrid ygrid variables
18	creating 1st solid
19	creating shadow lines
20	creating 2nd solid
21	assigning REAL values to variables
22	Graph changed by user
23	Graph changed by user
24	shadow line 2
25	north point
26	Graph changed by user
27	building2
28	Graph changed by user
29	Location SET
30	building2 height v
31	building 2 shadows
32	Building point created
33	Building 3 created
34	variable changes
35	Consolidation of 4 transactions
36	building5
37	Graph changed by user
38	shadows on b4
39	Consolidation of 4 transactions
40	color building5
41	Consolidation of 10 transactions
42	Graph changed by user
43	Graph changed by user
44	Graph changed by user
45	Graph changed by user

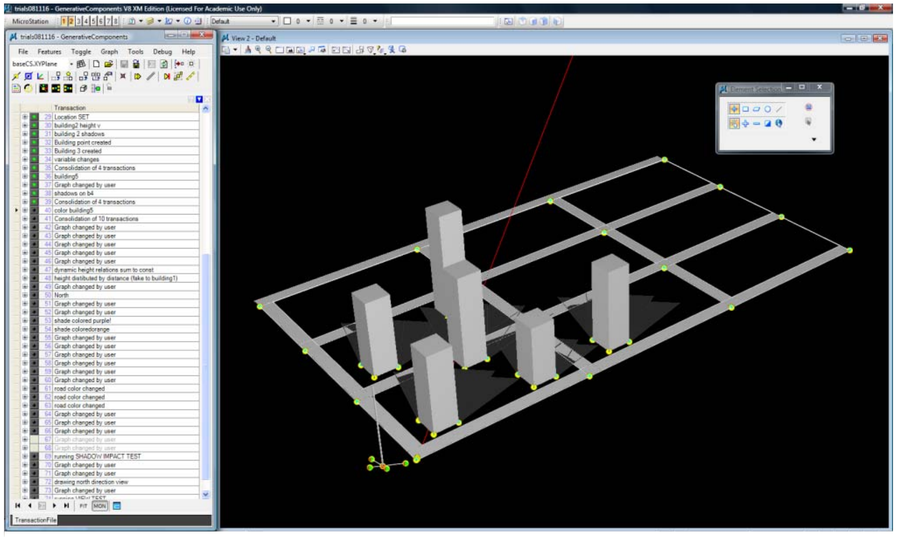
6.10

Buildings 4 is created and positioned relative to Building 2 according to the shading impact rule.

Building4.Ytranslation = Building2.Ytranslation

Building4.Xtranslation = Building2.Xtranslation + Building2. Side + SL2

```
transaction modelBased "dynamic height relations sum to const"
{
    feature Building GC.Solid
    {
        Zdimension                  = h4+h_var;
    }
    feature Building2 GC.Solid
    {
        Zdimension                  = h2+h_var;
    }
    feature Building3 GC.Solid
    {
        Zdimension                  = h3+h_var;
    }
    feature Building5 GC.Solid
    {
        Zdimension                  = h5+h_var;
    }
    feature Building6 GC.Solid
    {
        Zdimension                  = h6+h_var;
    }
    feature h1 GC.GraphVariable
    {
        Value                       = 55.0;
    }
    feature h2 GC.GraphVariable
    {
        Value                       = 80.0;
    }
    feature h3 GC.GraphVariable
    {
        Value                       = 75.0;
    }
    feature h4 GC.GraphVariable
    {
        Value                       = 28.05;
    }
    feature h5 GC.GraphVariable
    {
        Value                       = 60.0;
    }
    feature h6 GC.GraphVariable
    {
        Value                       = 77.0;
    }
    feature h_sum GC.GraphVariable
    {
        Value                       = 460;
        RangeStepSize               = 0.0;
    }
    feature h_var GC.GraphVariable
    {
        Value                       = (h_sum-(h1+h2+h3+h4+h5+h6))/6;
        RangeStepSize               = 0.0;
    }
    feature solid01 GC.Solid
    {
        Zdimension                  = h1+h_var;
    }
}
transaction modelBased "height distibuted by distance”
{
    feature Building GC.Solid
    {
        Zdimension                  = h4+h_var*1.5;
    }
    feature Building3 GC.Solid
    {
        Zdimension                  = h3+h_var*.5;
    }
    feature Building6 GC.Solid
    {
        Zdimension                  = h6+h_var*.5;
    }
    feature h1 GC.GraphVariable
    {
        Value                       = 75.0;
    }
    feature h3 GC.GraphVariable
    {
        Value                       = 85.0;
    }
    feature h5 GC.GraphVariable
    {
        Value                       = 30.0;
    }
}
```

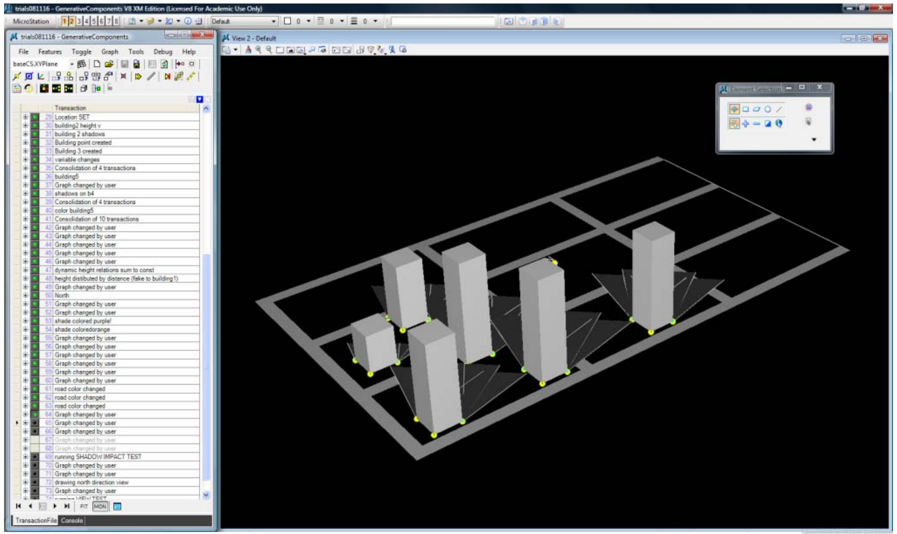

trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
Default
trials081116 - GenerativeComponents
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
View 2 - Default
FIT
MON
TransactionFile

Transaction
29 Location SET
30 building2 height v
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user

6.11

Buildings 5 and 6 are created according to the same positioning rules.

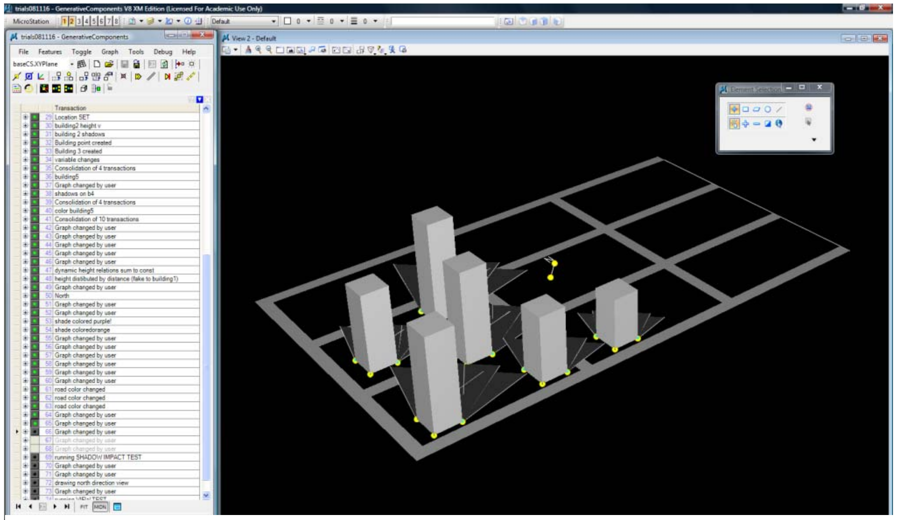
Height relationships are established between the buildings. *Variables* **h1, h2, h3, h4, h5, h6** sum to a fixed value to follow the fixed density rule. Changes in any of the building’s heights reflect back to a non-linear configuration change of other building heights and positions, following height and shadow rules.


trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
Default
trials081116 - GenerativeComponents
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
View 2 - Default
FIT
MON
TransactionFile
Console

Transaction
29 Location SET
30 building2 height v
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user

6.12

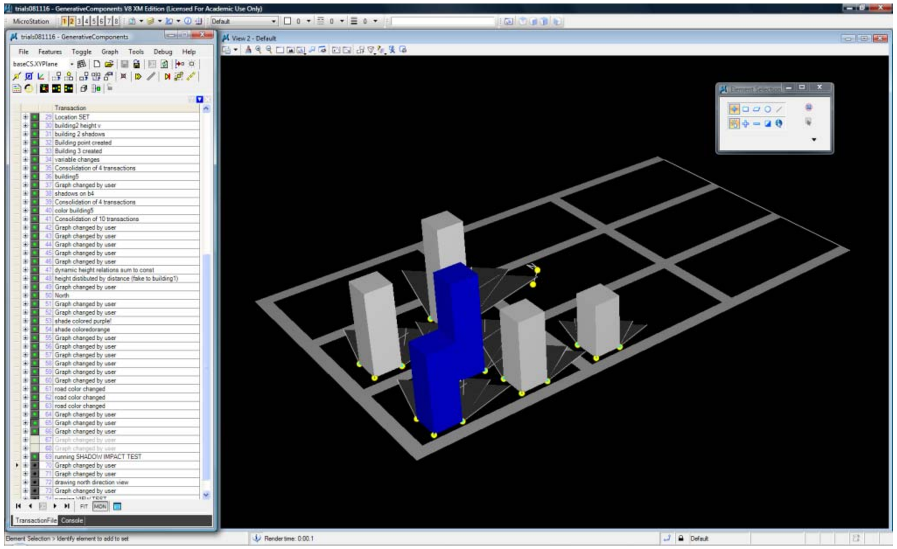
```
transaction script "POSITION TEST"
{
    if ( building5_a.YTranslation > point07[0][1].Y)


                                    {

                                        b5color=1;
                                        color=1;
                                        h1=h1max;

                                    }
                if ( building6_a.YTranslation > point07[0][2].Y)


                                    {

                                        color=3;


                                    }
}
```


trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
trials081116 - GenerativeComponents
File Features Toggle Graph Tools Debug Help
baseCS.XYPlane
View 2 - Default
Default
FIT MDN

```
Transaction
29 Location SET
30 building2 height v
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user
74 running VIEW TEST
```

6.13

A positioning test reconfigures the building's position, and accordingly the maximum allowable heights of other buildings if an initial model configuration violates positioning rules. In this case, building 5's initial position violates street clearance. The system adjusts its position and the max height for building1.

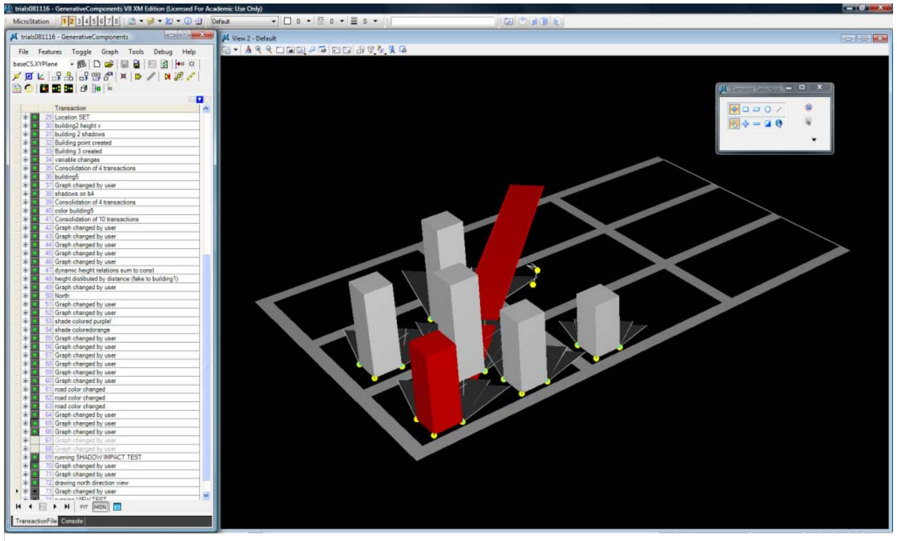

trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
trials081116 - GenerativeComponents
File Features Toggle Graph Tools Debug Help
baseCS.XYPlane
View 2 - Default
Default
FIT MDN
TransactionFile Console

```
Transaction
29 Location SET
30 building2 height v
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user
74 running VIEW TEST
```

Element Selection > Identify element to add to set
Render time: 0.00.1

6.14

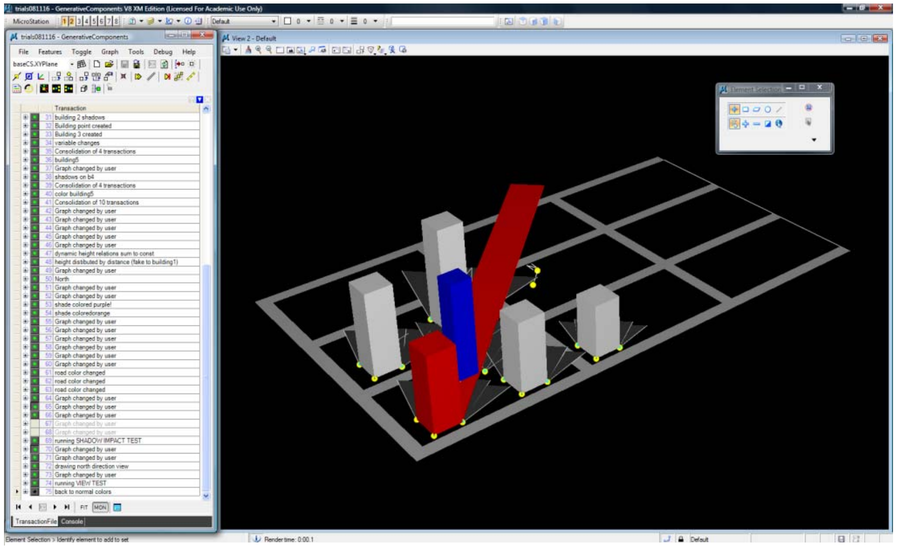
```
transaction script "running VIEW TEST"
{
    if ( projectb5.X < building5_b.XTranslation)
           {
                if ( projectb5.Y > building5_b.YTranslation)
                         {
                         if ( projectb5.X >
building5_a.XTranslation)
                                {
                                      if ( projectb5.Y <
building5_a.YTranslation)
                                         {
                                             b5color = 1;

                                             Building5.Xdimension=side-
6;
                                             Building5.Ydimension=side-
6;
                                         }
                                  }
                        }
           }
}
```


trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
Default
trials081116 - GenerativeComponents
View 2 - Default
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
FIT
MDN
TransactionFile
Console

Transaction
29 Location SET
30 building2 height v
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user

6.15

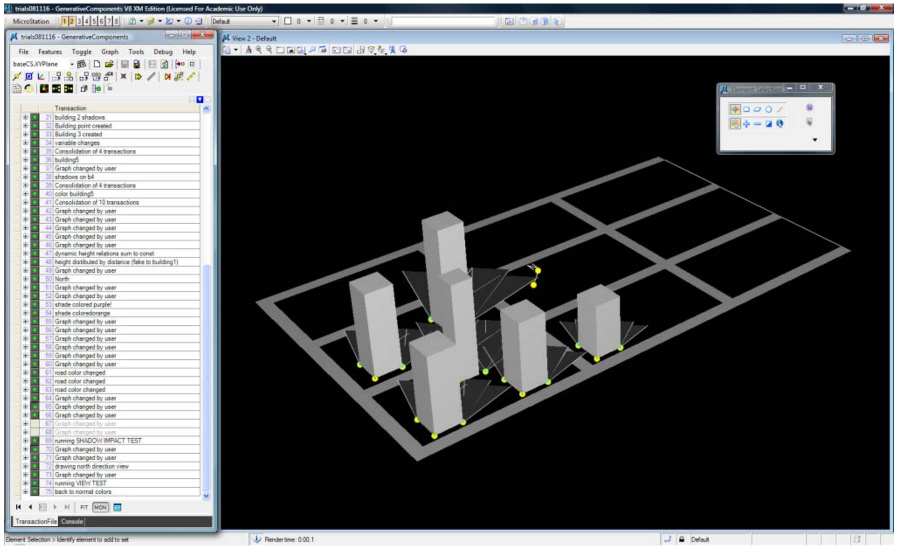
A view-blocking test projects lines towards the north direction. If a building blocks other buildings' north views, there is a decrease in its floor plate area to minimise the blocking impact.



Transaction
31 building 2 shadows
32 Building point created
33 Building 3 created
34 variable changes
35 Consolidation of 4 transactions
36 building5
37 Graph changed by user
38 shadows on b4
39 Consolidation of 4 transactions
40 color building5
41 Consolidation of 10 transactions
42 Graph changed by user
43 Graph changed by user
44 Graph changed by user
45 Graph changed by user
46 Graph changed by user
47 dynamic height relations sum to const
48 height distibuted by distance (fake to building1)
49 Graph changed by user
50 North
51 Graph changed by user
52 Graph changed by user
53 shade colored purple!
54 shade coloredorange
55 Graph changed by user
56 Graph changed by user
57 Graph changed by user
58 Graph changed by user
59 Graph changed by user
60 Graph changed by user
61 road color changed
62 road color changed
63 road color changed
64 Graph changed by user
65 Graph changed by user
66 Graph changed by user
67 Graph changed by user
68 Graph changed by user
69 running SHADOW IMPACT TEST
70 Graph changed by user
71 Graph changed by user
72 drawing north direction view
73 Graph changed by user
74 running VIEW TEST
75 back to normal colors

Element Selection > Identify element to add to set
Render time: 0:00.1

6.16


trials081116 - GenerativeComponents V8 XM Edition (Licensed For Academic Use Only)
MicroStation
trials081116 - GenerativeComponents
File
Features
Toggle
Graph
Tools
Debug
Help
baseCS.XYPlane
View 2 - Default
FIT
MON
TransactionFile
Console
Default

	Transaction
31	building 2 shadows
32	Building point created
33	Building 3 created
34	variable changes
35	Consolidation of 4 transactions
36	building5
37	Graph changed by user
38	shadows on b4
39	Consolidation of 4 transactions
40	color building5
41	Consolidation of 10 transactions
42	Graph changed by user
43	Graph changed by user
44	Graph changed by user
45	Graph changed by user
46	Graph changed by user
47	dynamic height relations sum to const
48	height distibuted by distance (fake to building1)
49	Graph changed by user
50	North
51	Graph changed by user
52	Graph changed by user
53	shade colored purple!
54	shade coloredorange
55	Graph changed by user
56	Graph changed by user
57	Graph changed by user
58	Graph changed by user
59	Graph changed by user
60	Graph changed by user
61	road color changed
62	road color changed
63	road color changed
64	Graph changed by user
65	Graph changed by user
66	Graph changed by user
67	Graph changed by user
68	Graph changed by user
69	running SHADOW IMPACT TEST
70	Graph changed by user
71	Graph changed by user
72	drawing north direction view
73	Graph changed by user
74	running VIEW TEST
75	back to normal colors

Element Selection > Identify element to add to set

Render time: 0:00.1

6.15

Figure 6.15 represents an instance of a valid height and positioning configuration according to rules of height, shading impact, and view blocking.

Conclusion:

This exploration displays a responsive approach to city planning and explores a method to establish a dynamic approach to city planning. I believe it is successful in regulating the existing and future relationship between buildings, and providing a dynamic system that controls the allowable locations and forms of buildings in an urban scale. The output of this exploration can be used as an application for city planners to map out the existing buildings in a particular neighbourhood and set the planning rules guiding the regulations of both existing and new buildings.

The system proves its potential for application on a variety of references. In principal, criteria like costs and property values could be integrated using the same logic of system construction.

On a social perspective, buildings exist in dynamic reference to their context. The proposed planning system provides methods to new planning and regulation processes that would start to acknowledge and integrate this dynamic existe

[10] http://vancouver.ca/commsvcs/guidelines/pol&guide.htm#dd
Accessed: 28/10/2008, 4:32 PM

7 Conclusions

Conclusions

This thesis applies parametric design techniques to a series of design explorations at scales ranging from residential unit interiors to cityscape configurations. The objective is to establish a responsive process of design and regulation, where programmatic and context-sensitive criteria set the rules and drive the process of design.

A rule-based approach to design was developed through the understanding of emergence logic and its potential to solve problems of rising complexity, like addressing programmatic concerns in designs that reflect and respond to larger sets of interrelated social and environmental references. The design explorations demonstrate an increasing complexity of design problems. It is possible that all four explorations be incorporated with each other. For instance, 'Space Configurations' could become an integral part of the 'Tower Formation' exploration, which in turn could be used inside the 'City Configurations' exploration to achieve added complexity and detail.

It is important to note the studies and explorations carried out in this thesis do not propose finalised design solutions, as much as they display a method to approach complex design problems in computational environments. This thesis demonstrates a proof of concept, where parametric tools are compared to earlier approaches of form-making and are found to provide the potential to address a complex set of social and programmatic criteria in a responsive and interactive design setting at a variety of scales and purposes. Previous digital techniques used by architects were found to focus on the ambiguous generation of formal complexity without reflecting the outside references essential for creating architecture.

A key aspect in the use of the proposed approach to design is the selection of rules and the translation of design problems and their references into components, parameters, and dependencies inside the parametric model. In principal, if the right parameters are selected, parametric design techniques can address, coordinate, and control various additional concerns and references. It is only a matter of added complexity and devotion. Parametric rule-based design relies on the intuitive judgment of the designer in the method of creating the responsive system. Exclusions, limits, and constraints need to be defined in order to set the allowable range of system possibilities. Beyond those limits, chaotic solutions may arise.

The rules for the system are explicit and the design assumptions are defined prior to starting design. Parametric tools display a complex method of storing explicit design rules through the design process. It can be argued that this is not possible with the human mind's computational abilities. The design explicitness creates a fair and closed world inside the parametric system, which is only free to respond to the pre-decided references. It has to be understood that, unlike CAD modeling, the investment is not so much in the geometries created, but in the rules, relationships, and dependencies that define how the geometries inside the parametric interact and respond to their references. Once these are set, parametric techniques demonstrate an interactive atmosphere in the design process, and provide potential to address added design complexities.

The final decision of the designed product is divided between the architect (designer), the user (client), and the parametric system involved. This approach to design is subjective in the way a designer sets the rules for the program to execute, however the designer could open the system to user input. This input could trigger changes in value weights, parameters, or references and would therefore trigger alterations in the overall design.

> *" The architectural field's current use of the parametric has been superficial and skin-deep, may be importantly so, lacking of a larger framework of referents, narratives, history, and forces."* (Meredith 1)

The increasing use of digital techniques in design clearly exhibits the architects' temptation to develop complex formal solutions. However, this thesis works towards a second phase in the use digital techniques to design architecture. It strives to display the ability to deploy parametric design techniques in designs that are responsive to the particular conditions of context, user functions, and program based on a complex set of interrelated natural and social references specific to urban housing.

Bibliography

aadrl.net. 2 4 2009 <http://www.aadrl.net/>.

Aish, Robert. "Positioning of GenerativeComponents." GenerativeComponents Version 08.09.04.76 Help . 2007.

Benjamin Aranda, chris Lasch. Tooling. New York: Princton Architectural Press, 2005.

Bernard Tschumi, Irene Cheng. The State of Architecture at the beginning of the 21st century. New York: The Monacelli Press, 2003.

Castle, Helen. "Emergence in Architecture." AD Wiley Academy 2004.

Chu, Karl. "Metaphysics of Genetic Architecture and Computation." AD Wiley Academy Vol.76 no 4 2006: 38-45.

City of Vancouver, Planning Department, Statistics and Information. 28 10 2008. <http://vancouver.ca/commsvcs/planning/stats.htm>.

Cole, Ray. "Environmental Systems and Control I." Course Notes. 2005.

DeLanda, Manuel. "Deleuze and the Use of Genetic Algorithms in Architecture." AD Wiley Academy January 2002: 9-12.

Gilles Deleuze, Felix Guattari. A Thousand Plateaus. University of Minnesota Press, 1987.

Hensel, Michael. Emergence: Morphogenetic Design Strategies. Academy Press, 2004.

Holland, John. Emergence: From Chaos to Order. Basic Books, 1998.

Landa, Manuel De. A Thousand Years of Nonlinear History. New York: Zone Books, 2000.

Leach, Neil. "Digital Morphogenesis: A New Paradigmatic Shift in Architecture." Archithese 2006: 44-49.

Martin Hemberg, Una-May O'Reilly. "Geometry as a Substitute for Structural Analysis in Generative Design Tools." (n.d.).

Menges, Achim. "Morpho-Ecologies: Approaching Complex Environments." Emergence - Morphogenetic Design Strategies, AD Wiley Academy 2004.

Michael Hensel, Achim Menges, Michael Weinstock. "Emergence in Architecture." AD Wiley Academy 7 2004.

—. Emergence: Morphogenetic Design Strategies. AD Wiley Academy, 2004.

—. "Frei Otto in Conversation with the Emergence and Design Group." AD Wiley Academy 2004: 18-25.

Michael Meredith, Mutsuro Sasaki, Aranda-Lasch. From Contril to design: Parametric/Algorithmic Architecture. Actar, 2008.

Michel Hensel, Achim menges, Michael Weinstock. "Techniques and Technologies in Morphogenetic Design." AD Wiley Academy Vol.74 No 3 May/June 2004.

Michel Weinstock, Achim Menges, Micheal Hensel. "Fit Fabric: Versatility through Redundancy and Differentiation." AD Wiley Academy 2004: 40-47.

Picon, Antoine. "Architecture and the Virtual towards New Materiality." Praxis 6: New Technologies/ New Architecture 2004: 114-121.

—. "Toward a Well-Tempered Digital Design." Harvard Design Magazine Fall 2006/ Winter 2007: 77-83.

Rahim, Ali. Catalytic Formations: Architecture and Digital Design. Taylor & Francis, 2006.

—. Contemporay Techniques. Architectural Design Vol. 72, 2002.

Rocker, Ingeborg M. "When Code matters." AD Wiley Academy 2006: 16-25.

smartgeometry.org. 15 2 2007 <http://www.smartgeometry.org/>.

Spuybroek, Lars. NOX. Thames & Hudson, 2004.

Terzidis, Kostas. Algorithmic Architecture. Architectural Press, 2006.

Thompson, D'Arcy Wentworth. On Growth and Form. Cambridge University Press, 1942.

Una May O'Reilly, Martin Hemberg, Achim Menges. "Evolutionary Computation and Artificial Life in Architecture: Exploring the potential of Generative and Genetic Algorithms as Operative Design Tools." AD Wiley Academy Vol.74 no 5 2004: 48-53.

Vancouver Views: Council Approved View Cones. 18 10 2008 <http://vancouver.ca/commsvcs/views/viewcones/91.htm>.

Weinstock, Michael. "Morphogenesis and the Mathematics of Emergence." AD Wiley Academy 169 2004: 10-17.

Figures credits

All drawings, diagrams, and photographs in this thesis are produced by Yehia Madkour, except as noted below:

Figure 1.2 Game Tree for Tic-Tac-Toe
Holland, John. Emergence: From Chaos to Order. Basic Books, 1998, 37

Figure 2.1 Son O House
Spuybroek, Lars. NOX. Thames & Hudson, 2004.

Figure 2.2 Son O House Process
Spuybroek, Lars. NOX. Thames & Hudson, 2004.

Figure 2.3 Log Cabin Process 1
Benjamin Aranda, chris Lasch. Tooling. New York: Princton Architectural Press, 2005,23,25.

Figure 2.4 Log Cabin Process 2
Benjamin Aranda, chris Lasch. Tooling. New York: Princton Architectural Press, 2005,29,30.

Figure 2.5 Pneumatic Strawberry Bar Process
http://www.achimmenges.net/achimmenges_pl04_DesignExper.swf
Accessed 13 4 09