

**Embedding Methods for Massing and Detail Design
in Computer Generated Design of Skyscrapers**

By

Shouheng Chen

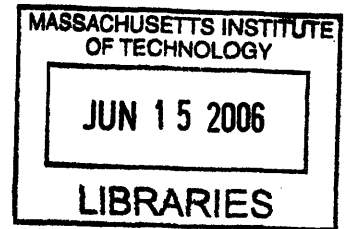
Master of Architecture (2001)
University of Toronto

Bachelor of Architecture (1997)
Shenzhen University

Submitted to the Department of Architecture
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Architecture Studies
at the
Massachusetts Institute of Technology

June 2006

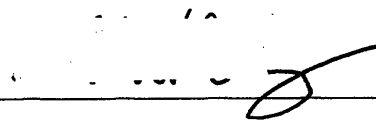


© 2006 Shouheng Chen. All rights reserved


ARCHIVES

The author hereby grants to MIT permission to reproduce
And to distribute publicly paper and electronic copies of this
Thesis document in whole or in part.

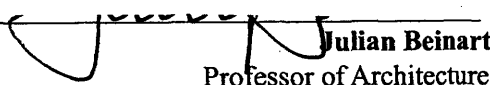
Signature of Author: _____


Shouheng Chen
Department of Architecture
June 2006

Certified by: _____


Takehiko Nagakura
Academic Head, Design and Computation
Associate Professor of Design and Computation

Accepted by: _____


Julian Beinart
Professor of Architecture
Chairman, Department Committee on Graduate Students

Thesis Readers:

Axel Kilian
Postdoctoral Associate
Department of Architecture
Massachusetts Institute of Technology

Martin Bechthold
Associate Professor of Architecture
Harvard Graduate School of Design

Edward Dionne
Senior Associate, Pelli Clarke Pelli Architects
(Formerly Cesar Pelli and Associates)

Embedding Methods for Massing and Detail Design in Computer Generated Design of Skyscrapers

By

Shouheng Chen

Submitted to the Department of Architecture
on May 25, 2006 in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Architecture Studies

Abstract:

This thesis proposes a new digital system to construct the massing and details of skyscrapers. It extracts underlying rules and design conventions from significant projects in contemporary skyscraper design practice. These rules and conventions are translated into digital data and embedded in a system. The thesis demonstrates how to use this system to reconstruct original designs as well as to generate new ones by means of transformation rules. It takes examples from the built skyscraper projects of Cesar Pelli and Associates as well as Norman Foster and Partners, and embeds their conventions and components to illustrate an implementation of such a system.

In contemporary skyscraper design, sophisticated computer models are constructed in advanced engineering systems for the use of engineering analysis, but they contribute very little to the conceptual design of skyscrapers. The goal of this thesis is to propose embedded methods as an alternative approach and to develop a digital system that can both handle complex forms and enable architects to work more efficiently in the early stages of the design process. The intention behind building such a system is to relieve architects from the repetitive work that is required by conventional CAD systems as well as to allow them to carry their previous expertise--well-established stylistic conventions and approved components--into the design of new skyscrapers.

Thesis Advisor: Takehiko Nagakura
Title: Associate Professor of Design and Computation

Acknowledgements:

I want to thank the following people for making my experience at MIT significant:

Takehiko Nagakura -- for providing excellent guidance throughout my thesis and for the productive collaboration in the Gyeonggi-do Jeongok Prehistory Museum competition. I thank him for awarding me teaching assistantships and for giving me the opportunities to participate in his incredible research projects.

Axel Kilian -- for his great support and inspiration and for his help on developing my thesis. I thank him for always being around to help.

Martin Bechthold -- for his insight and enthusiasm in my thesis topic and for his participations in numerous thesis discussions.

Edward Dionne -- for his extremely valuable practical input into my thesis and for sharing with me his knowledge of skyscraper design and construction.

and my colleagues at MIT, for their willingness to discuss each other's work. I have truly learned a lot from these extremely valuable discussions.

“Code is character. Code is the law.”

-- William J. Mitchell

Table of Contents:

Abstract

Acknowledgements

Chapter 1: Introduction

- 1.1 Overview**
- 1.2 Problem Statement**
- 1.3 Purpose**
- 1.4 Scope**

Chapter 2: Background

- 2.1 Historical Shortcomings of CAD Systems**
- 2.2 Complexity in Current Designs of Skyscrapers**
- 2.3 Knowledge-Based Systems in Architecture**
- 2.4 Related Work: Twisted Skyscraper**

Chapter 3: Case Studies

- 3.1 Non-Structural Skin Projects**
 - 3.1.1 Impact of Cesar Pelli in Curtain Wall Design
 - 3.1.2 Project One: Hong Kong International Finance Center
 - 3.1.3 Project Two: Petronas Tower
 - 3.1.4 Project Three: Cira Tower
- 3.2 Structural Skin Project**
 - 3.2.1 Norman Foster and Structural Diagrids
 - 3.2.2 Project Four: Hearst Corporation

Chapter 4: Computational Approaches

- 4.1 Structure of the Program**
- 4.2 Algorithmic Methods for Constructing Massing**
 - 4.2.1 Profile Finding Method
 - Typical Plan Profile Construction
 - Level Finding
 - Transformation Rules
 - 4.2.2 Point Finding Method

4.3 Algorithmic Methods for Constructing Detail

4.3.1 Associative Relationships between Position Points
and Adjacent Points

4.3.2 Component Construction

4.4 User Interface

Chapter 5: Experiments

5.1 Massing

5.1.1 Types of Massing

5.1.2 Constructing Massing in a Site Context

5.2 Detail

5.2.1 Types of Details

5.2.2 Constructing Details on a Selected Surface

5.2.3 Constructing Details on Selected Levels

5.2.4 Construct Details in Different Resolution Settings

5.3 Incorporation of Massing and Detail

New Design One: Drum-Shaped Tower

New Design Two: New Twisted Tower

Chapter 6: Conclusion and Future Work

**6.1 Conclusion--Towards a New Digital System
for Skyscraper Design**

6.2 Future Work--To Expand the Capacity of the System

Appendices

Appendix A: VB.Net

Appendix B: Rhinoscript

Appendix C: Table of Figures

Bibliography

Chapter 1: Introduction

1.1 Overview

My passion in using computer programs to generate three dimensional models and to find design solutions for particular design problems convinced me to write this thesis.

The increasingly formal, technical, and ecological complexities in skyscraper design and construction is the catalyst for the development of new digital systems. These new systems have great impact on today's engineering advancement, typically that of structural systems of skyscrapers. Sophisticated computer models, generated by these systems, are efficient and effective ways to analyze and evaluate complex design situations.

Because these systems are primarily developed by engineers and mainly used in engineering applications, they contribute very little to the early stages of the conceptual design of skyscrapers. Architects are still using conventional CAD systems that are far less powerful than those of engineers. These conventional systems are very ineffective in handling repetitive and non-standard components, which are the major features of current skyscraper design and construction.

The goal of this thesis is to propose embedded methods as an alternative approach to construct sophisticated

computer models for architectural design purposes. The core idea is to provide architects with a digital system that can both handle complex forms (the massing) and achieve efficiency in the early stage of the design process. The intention behind building such a system is to relieve architects from the repetitive work that is required by conventional CAD systems as well as to allow them to carry previous expertise--well-established stylistic conventions and approved components (the detail) --into the new design of skyscrapers.

The development of such a new system demands an in-depth understanding both of what the well-established conventions and the approved components are and of how they can be embedded into the system. In order to do so, the thesis uses Cesar Pelli and Norman Foster's projects as a point of departure and investigates techniques that these two leading designers have developed through decades of practice. Then, these techniques are translated into digital data and embedded into the proposed system. In the process of developing such a system to re-construct the original design, the thesis identifies the underlying rules for the re-construction and demonstrates a way to expand the capacities of the system to construct new skyscrapers.

1.2 Problem Statement

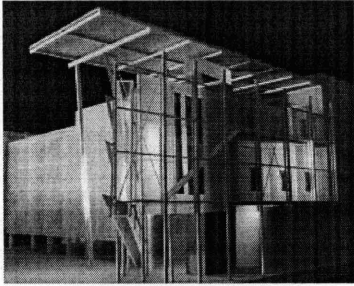


Figure 1: Dundas Entertainment Center shows the construction of highly tectonic, de-constructive architectural surfaces.

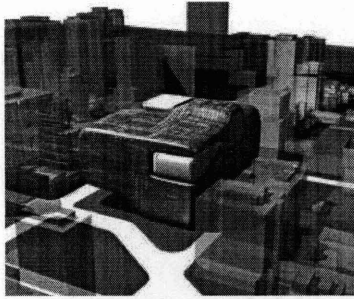


Figure 2: Informational & Mediatized Architectural Surfaces shows the construction of folded and mediatized architectural surfaces

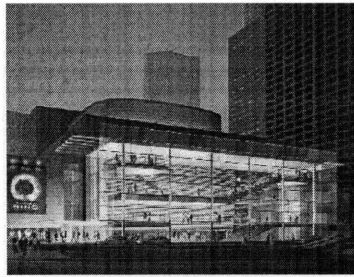


Figure 3: The Canadian Opera House shows the construction of transparent structural glass (with Diamond and Schmitt Architects Inc.)

My previous academic and practical work has varied in program and complexity, but all of my design share one common notion: architectural surfaces function as a device to blur the boundaries between exterior and interior and promote visual interaction between users and the building. In the Dundas Entertainment Center (Figure 1, 1999), I interpret this idea by constructing highly tectonic, de-constructive skins that are either structural or non-structural skins. The level of interaction is controlled by the degree of transparency and translucency of the skins. The sense of arrival at the main lobby on the third floor is defined by a pathway wrapped by double layered skins with signs and projections. In my second project, Informational and Mediatized Architectural Surfaces (IMAS) (Figure 2, 2001), I demonstrate the idea very differently. The skins are folded to reflect the internal and external forces. These forces respond to different sources, such as gravity, programs, and a sense of approaching. Reacting to gravity, a certain area of the surface is folded down to allow the surface to attain strength until it reaches the capacity to hold the objects sitting on top of it. The folding of the surface in the entry area makes up a welcoming gesture for people approaching the building. An even more important feature of the surfaces is the projected images that cover the entire building. These images, on the one hand, merge the building into its vibrant urban setting in downtown shopping area in Toronto; on the other hand, they reveal the contents of the building to the outside.

The Canadian Opera House (Figure 3, 2004), my currently completed project in the financial and theatre district of downtown Toronto, shows the idea through the highly transparent structural glass in the façade at University Avenue. This structural glass façade links the city and the opera and highlights the project as a lit-lantern in the center of the city. In my most recent projects, the Tulsa Regional Events and Conventions Center (Figure 4, 2005), surfaces are treated in a different way. These surfaces are composed by layers of zinc-clad curvilinear surfaces. They are intended to identify the building as an architecturally significant icon in the City of Tulsa as well as to inform people about the path of approach to the building. These surfaces are laid out and scaled in such a way that people will find intimacy in this gigantic building upon entry.

In parallel to the interest in surfaces, my interest in architecture is also about tools that construct these surfaces. I used physical modeling as a means of exploration in the Dundas Entertainment Center (Figure 5). In 1999, at the time Rhino was first introduced to me by Professor Shane Williamson at the Faculty of Architecture, Landscape and Design, University of Toronto. I decided to proceed differently in the IMAS project and use Rhino in the design exploration (Figure 6). I was stunned by the power and ease of Rhino in constructing complicated forms. When I was developing the Canadian Opera House at Diamond and Schmitt Architects Inc. (DSAI) in Toronto, I was very pleased by my ability in handling both physical and digi-

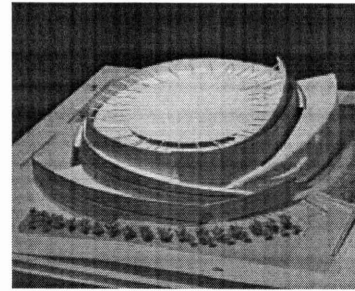


Figure 4: The Tulsa Arena shows the constructs of overlapping curvilinear zinc-clad surfaces (with Cesar Pelli and Associates)

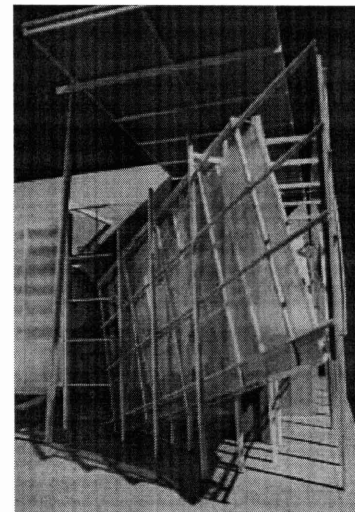


Figure 5: Dundas Entertainment Center shows the use of physical modeling in design exploration.

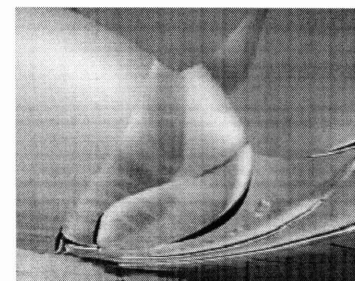


Figure 6: Reflected Panels in the Auditorium of the Canadian Opera House Study models were constructed in Rhino. This image shows the actual built reflected panels.

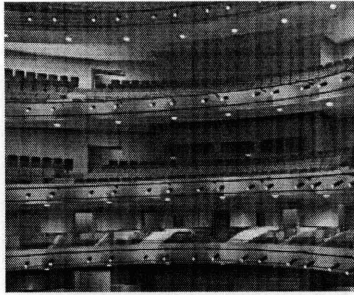


Figure 7: Balcony Front in the Auditorium of the Canadian Opera House Study models were constructed in Rhino. The balcony front is covered by metal panels and these panels differ in angles and curvatures.

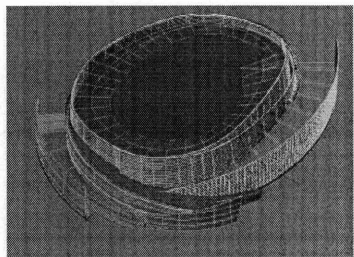


Figure 8: The Tulsa Arena shows 3D model was constructed in Rhino to understand the spatial relationship of overlapping curvilinear walls.

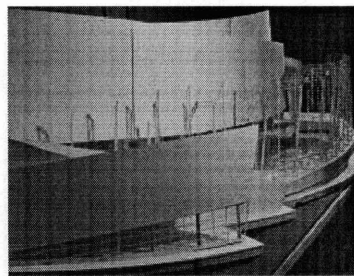


Figure 9: The Tulsa Arena shows a large-scaled physical model was constructed, showing the assembly of overlapping zinc-clad walls and curtain wall.

tal modeling for such a complex practical project. Both physical and digital modeling were extensively tested in the design process and made significant contribution to the flow of the project. However, the Opera project also revealed the limitations of both tools. In a relatively complex project like the Opera (Figure 6 & 7), both physical and current digital tools were not efficient under many circumstances. For example, the tilted and curvilinear balcony front in the auditorium required intensive and repetitive work. In fact, most of the balcony components were similar but only varied in tilted angles and positions. Neither physical nor present digital tools were able to release me from the “meaningless” manual operations of constructing uncountable similar components. Similar situation occurred in the back walls of the auditorium and in some other areas on the building. The mission was accomplished, but I was in doubt about the tools we were using. If the Opera was the wake-up call for new tools, the Tulsa Regional Events and Conventions Center (Figure 8 & 9) was the proof of this emergent need. Besides all the problems I have encountered in the Opera, intensive and repetitive work on uncountable similar components, I was also highly stressed by being afraid of losing accuracy on constructing those tilted and curvilinear walls in the process of manual operation. It was not until I realized that our associate architects and engineers were taking the responsibility for actually assembly these walls. At the very least, I knew that there was someone who would check the models a few more times to make sure that no mistakes were made.

1.3 Purpose

Having a retrospective review of my previous experiences on using different tools in design, I realize the increasing complexity from one project to the other and I also understand the advantages and disadvantages of tools that I have been using. By using physical models, designers are limited in reaching a certain level of complexity. Conventional CAD systems have been proven to be effective in handling relatively complex forms, but their limitations are also apparent: they require intensive manual operation and repetitive work. Designers become the sole operator of these systems. In fact, taking the Tulsa Arena for example, overlapping metal clad curvilinear surfaces were constructed very similarly. With the available tools in the office, we had to build every individual surface. Any adjustment or change to the design required a lot of efforts to rebuild these surfaces. Conventional CAD systems are very cumbersome in this case. Complexity in design is one of the major issues in current design trend. Figure 10, 11 & 12 show some built or conceptual projects of complex forms and tremendous numbers of non-standard components. Without using advanced systems, it will not be possible to build these projects.

Complexity in design is a concern, but it is not the only issue that matters in design. Typically for skyscraper design, there are design conventions and techniques having been developed over many decades. These conventions and techniques are extremely valuable, setting up

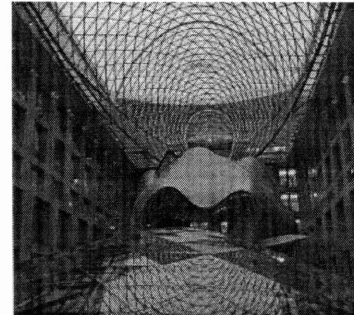


Figure 10: DG Bank by Frank Gehry shows triangulated complex surfaces - image taken from Branko Kolarevic's Digital Production

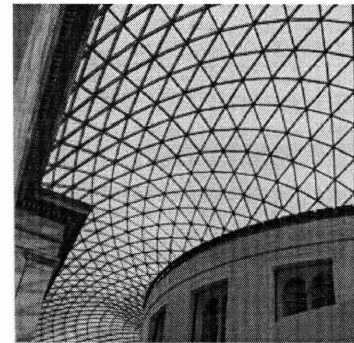


Figure 11: British Museum Great Court by Norman Foster shows triangulated toroidal surfaces - image taken from Branko Kolarevic's Digital Production

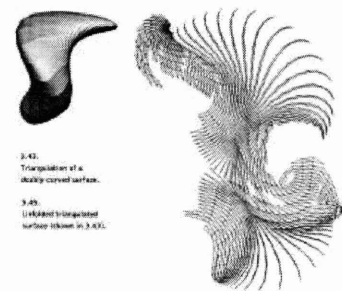


Figure 12: Triangulation of a doubly-curved surface - image taken from Branko Kolarevic's Digital Production

practical guidelines for designers. Taking Cesar Pelli for example, he has developed very profound systems for curtain wall design and construction. This system leads to the building of many significant skyscraper projects.

The purpose of this thesis is to target both issues of complexity and design conventions and techniques. The proposed system is attempting to provide a mechanism to handle complex forms and at the same time provides opportunities to embed well-established stylistic conventions and techniques into the system.

1.4 Scope

Skyscraper design is a very large topic and the design of a tower requires intensive cross-discipline cooperation between architects, structural engineer, mechanical engineer and many others. Also, many programs have been developed by engineer to analyze structural integrity and other performances of a tower. These programs have very powerful analytical functions and are able to provide analytical data to help architect understand design problems. The scope of this thesis is not to compete with these systems, but to propose a system that works in parallel with these engineering systems. It intends to provide a digital system to help architects construct sophisticated computer models in the earlier stage of conceptual design. These models could then be tested by engineering systems.

Chapter 2: Background

2.1 Historical Shortcoming of CAD Systems



Figure 13: The Sketchpad System by Ivan Sutherland
- image taken from Yehuda E. Kalay's Architecture's New Media

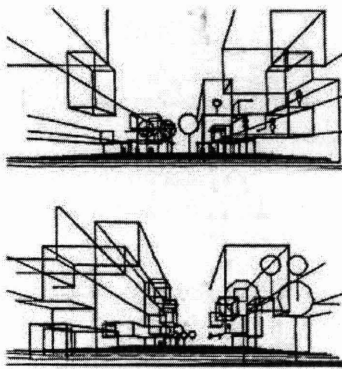


Figure 14: URBAN5
an early computer-aided urban planning system - images taken from Yehuda E. Kalay's Architecture's New Media

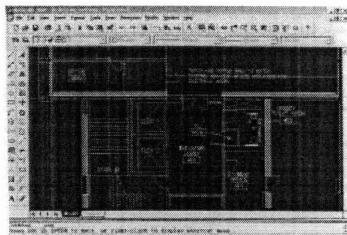


Figure 15: AutoCAD by AutoDesk
- image taken from Yehuda E. Kalay's Architecture's New Media

The advent of computers in the 1950s provided hope for many who looked for efficient and effective ways to produce variations of design intentions and to fabricate these variations for reviews and presentation purposes (Figure 13). Yet, the application of computer in architectural practices did not begin until the invention of early drafting and modeling systems in the 1970s (Figure 14).¹ With limited capacity and low efficiency, these earlier systems did not achieve the initial goals to draft in an intelligent way nor could they replace traditional drafting methods. The invention of “intelligent” systems in the 1980s kindled the hope again, but these “intelligent” -- knowledge-based – systems did not make a significant impact on architecture due to the difficulty of constructing comprehensive building-specific data and the reluctance of architects to share their practical knowledge with others.²

A breakthrough in CAD systems occurred in the 1990s. Two major factors accounted for this breakthrough: the globalization of the building industry and the advancement in computer software and hardware development. Economic growth and the use of personal computers also contributed to the rapid development of CAD systems. These were point-and-click systems (conventional CAD systems) (Fig-

1. Yehuda E Kalay, “Computing in Architectural Design” in Architecture's New Media. The MIT Press, Cambridge, Massachusetts. 2005. pp 64.

2. Ibid., 71

ure 15), capable of supporting professional drafting, modeling and renderings and greatly increasing efficiency and productivity in design. However, because point-and-click systems are designed to be used by wide range of users, they have limited capacity for architectural application. They required intensive manual operations and repetitive work from architects. Architects were the sole operators of these systems, but they could not work creatively by using these systems. As a result, many architects are still very resistant in using these conventional CAD systems in design today.

Conventional CAD systems were undeniably useful, and they have made design work more efficient, but they have not opened up new domains to the architectural imagination. In recent years, some cutting-edge projects, such as Frank O Gehry's Guggenheim Museum in Bilbao (Figure 16) and Norman Foster's Swiss Re in London, have been reinventing the use of CAD systems and providing new direction to the future. Many avant-garde researchers and designers are working towards establishing new systems (Figure 17) to handle the increasingly complex design problems.

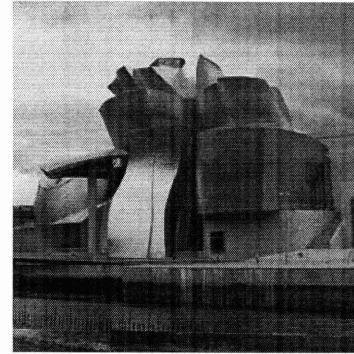


Figure 16: Guggenheim Museum by Frank Gehry
The image shows a curvaceous, free-form sculptural style - Gehry's signature style. - image taken from www.greatbuildings.com/buildings/Guggenheim_Bilbao.html

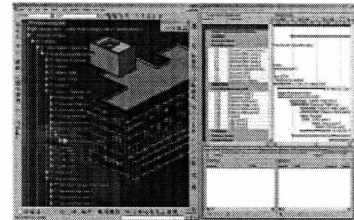


Figure 17: Digital Project by The Gehry Technology
A new digital system shows the use of building information modeling for design and construction - image taken from www.gehrytechnologies.com

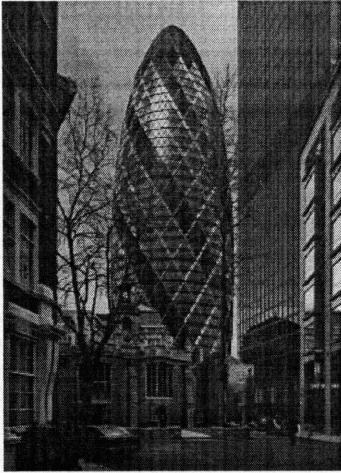


Figure 18: Swiss Re by Norman Foster (an aerodynamic and glazed shape tower in London - image taken from Guy Nordenson's Tall Buildings)

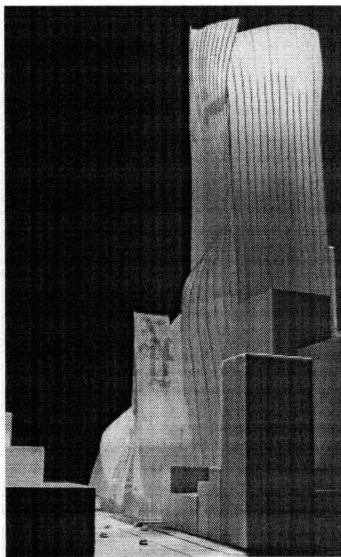


Figure 19: New York Times Headquarters by Frank Gehry/SOM (show double-curved and overlapping surfaces - image taken from Guy Nordenson's Tall Buildings)

2.2 Complexity in Current Designs of Skyscrapers

Because of the advancement in structural engineering and developments in curtain wall construction, architects today can essentially build as high as they like. The desire to build taller and taller skyscrapers is no longer a major concern for many skyscraper designers. Instead, it is the formal, technical, and ecological complexities of today's skyscrapers that challenge architects and lead innovation in the design of skyscrapers.

Many new skyscrapers such as the Swiss Re in London by Norman Foster (Figure 18) and the New York Times Headquarters by Frank Gehry/SOM (Figure 19) present an intention to construct complex exterior envelopes that are either curvilinear or multi-faceted skins. Unlike the facades of conventional skyscrapers that are the same, the facades of these new built skyscrapers provide dynamic views from the outside, giving a multi-dimensional reading of the building. These multifaceted skins reveal the complex functional programs inside the buildings. In this sense, the concept of creating complex forms reflects the practical requirements of skyscrapers in satisfying multi-functional programs. However, it is also valid to say that some architects see achieving complexity as a stylistic pursuit or as a deconstructive action against conventional design methodologies, and their design projects show these intentions. A good example of this is the Max Reinhardt Haus in Atlanta by Peter Eisenman.

From the technological point of view, the complexities of current forms provide both construction challenges and opportunities to open up new domains in skyscraper design. When the skins are faceted and twisted, components that compose the facades can no longer be standardized under current technologies. In order to deal with a lot of irregular and non-standard components, new standards and techniques are being developed. Also, new design systems are being constructed to help architects understand and evaluate complexities. An excellent outcome of this drive in technology is the Digital Project System, developed by Gehry Technology, a system that is intended to handle extremely complex design situations. It was initially used only by Frank Gehry, but now has been distributed for hundreds of users. To summarize, the pursuit of complexity has led to the establishment of new industrial standards as well as the development of new design systems. This is where the thesis is grounded.

2.3 Knowledge-Based Systems in Architecture

The objective of knowledge-based CAD systems is to change the focus from automating drafting to automating broad decision making and repetitive engineering task in the actual design process.³

Recently, knowledge-based systems have been widely used by electrical, mechanical, and acoustical engineering as well as in the aerospace and building construction industries. By using knowledge-based engineering systems, engineers are able to create, sustain and share design knowledge. With less effort in managing increasingly complex designs, engineers can better utilize knowledge for the design, the design process, manufacturing, and operating requirements. As a result, engineers can carry out new design projects more rapidly. Today, knowledge-based CAD systems provide opportunities for leading construction and manufacturing companies to dramatically increase productivity, improve quality, and at the same time dramatically reduce costs in their engineering services.

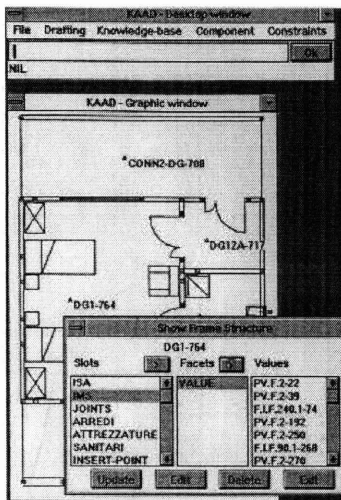


Figure 20: KAAD by G.Crrara et al. a framed-based knowledge representation system for the design of hospital - image taken from Yehuda E. Kalay's Architecture's New Media

In architecture, knowledge-based systems refer to graphics-oriented or object-oriented systems that handle not only shapes but also objects. In contrast to their counterparts in engineering, knowledge-based architectural systems have gone through a very slow development. However, we do see some progress. The very first knowl-

3. Cl Dym and RE Levitt, Knowledge-based systems in engineering. McGraw-Hill, New York, 1991.

edge-based computer-aided architecture system, World-View, was developed by a research group led by Yehuda Kalay at the University of Buffalo.⁴ WorldView addressed architectural objects such as walls, doors, and windows. It had the ability to control their geometric and non-geometric properties. Another system, KAAD (Figure 20), was developed in Italy by a group of researchers led by Gianfranco Carrara at the University of Rome.⁵ It was intended to be used in hospital design and was able to create architectural building and space components such as walls, doors, and rooms. It was also used to detect violations of rules that concerned the separation of clean and unclean areas, fire egress, and the placement of furniture. In the late 1990s, Building Design Advisor (BDA) was developed at the Lawrence Berkeley National Laboratory.⁶ This system was intended to support analysis of various energy-related building performance measures.

There are still other systems that have been developed since the late 1980s; however, none of these systems are popularly used in architectural practice. As I have mentioned in Chapter 1, the reason for this phenomenon is three-fold. First, most architectural offices lack powerful hardware and software supplies and are often unwilling to invest in them. Second, most offices lack a sufficient number of skilled architects, who can handle both design and computer programming at the same time. Third, most offices are reluctant to convert their design knowledge

4. Yehuda E Kalay, "Computing in Architectural Design" in *Architecture's New Media*. The MIT Press, Cambridge, Massachusetts. 2005. pp73.

5. *Ibid.*, 73

6. *Ibid.*, 73

into digital data for sharing with others. Architects are greatly concerned about losing their competitiveness by doing so.

The difficulty of development of knowledge-based systems in architectural application should not undermine their potential in architectural design. Understanding their success in engineering, I believe that these systems can make faster and faster progress following advancements in hardware and software development and the increasing open-mindedness of architects about the significance of these systems in opening up new domains in architectural design. By writing a knowledge-based system to construct some complex design of skyscrapers digitally, I intend to catalyze this process.

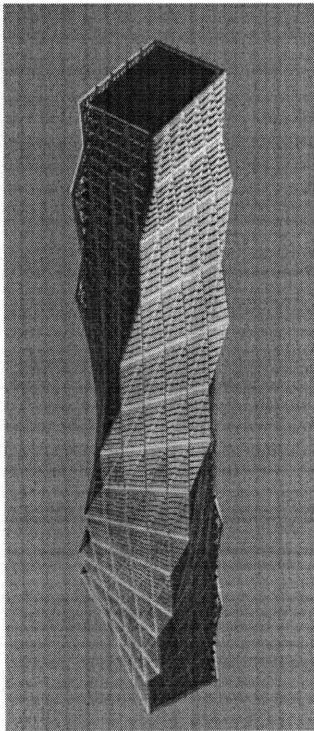


Figure 21: The Twisted Skyscraper shows the potential of computer programming in constructing facade details in skyscraper design.

2.4 Related Work: The Twisted Skyscraper

The Twisted Skyscraper (Figure 21) is my first project using computer a program for design. The program was written in AutoLISP and built through a progressive development. At first, the program was able to generate two-dimensional patterns and then three-dimensional shapes. Computational methods were developed to generate the diagrid structure of the tower. At last, its capacity had been expanded to construct three different types of details on the facades: vision panels, horizontal louvers, and perforated metal panels.

The program runs sequentially. It picks up plan out-

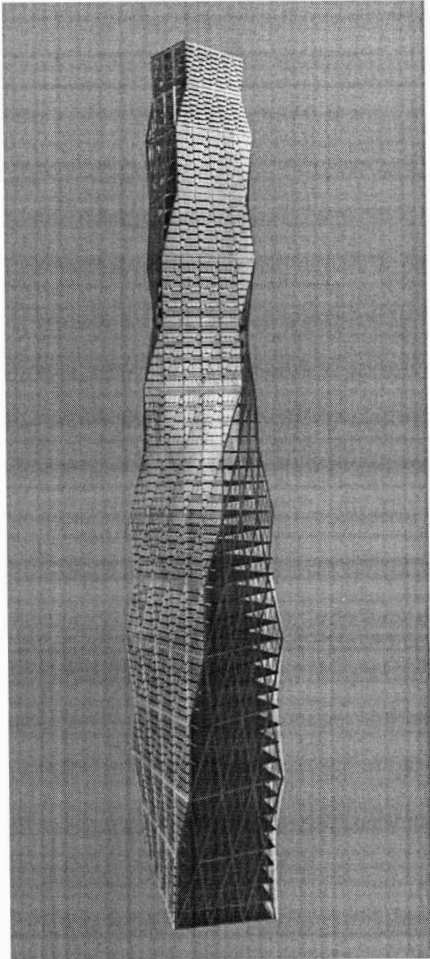
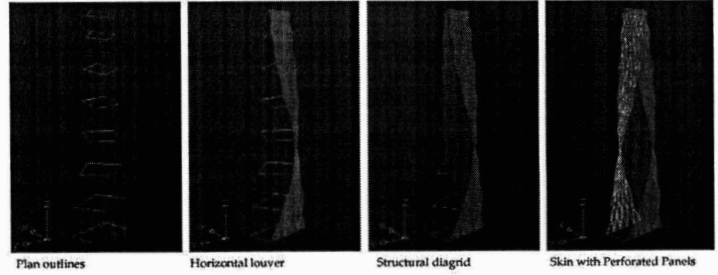


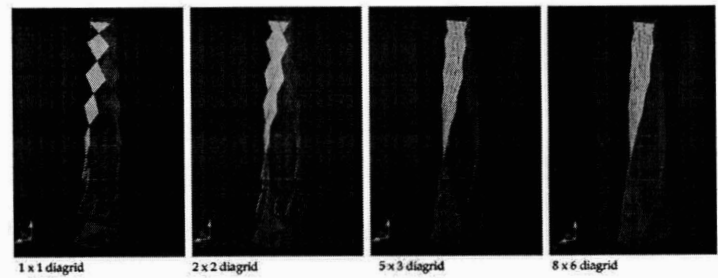
Figure 22: The Twisted Skyscraper Images shows the shape generation procedures. The program was written in AutoLISP and runs in AutoCAD.

Shape Generation:

Diagrams of shape generation:



New design options:



Procedures of shape generation:

1. Determine levels of diagrids (8 floors in height subdivided by number of floors);
2. Determine numbers of diagrids on each facade (6 to 8 meters in width);
3. Draw outlines of levels;
4. Run command "tower";
5. Select first column of outlines for "louver" generation, from bottom up;
6. Select second column of outlines for "frame" generation, from bottom up;
7. Select third column of outlines for "mesh" generation, from bottom up;
8. Select fourth column of outlines for "frame" generation, from bottom up.

lines on every eight floors (a practical standard for high-rise building with structural skins) and subdivides these outlines into a number of diagrids that are predefined in the program. Then, it generates solid diagrid tubes, on top of which secondary structural mullions are located. The program is responsive to the twisting nature of the tower: the orientations of the tower change as the tower rises. It produces horizontal louvers from the lower portions of its south elevation and gradually wraps around the west façade on top floors. A similar approach is taken on the north and east façades. Instead of louvers, either perforated metal panels or horizontal translucent panels are constructed to provide sun shading for the building (Figure 22).

This twisted and tapered tower has a lot of repetitive and non-standard components. It is this nature of complexity that presents the significant advantages of computer programming. The success of this exploration proves that using computer programming in design is an effective and efficient way to generate complex forms and yet maintains creativity and intuitive in the design process. I was greatly inspired by this project and it led to my thesis topic: to develop a fully functional and parameterized program to generate skyscrapers.

Chapter 3: Case Studies

Case studies of selected built skyscrapers were conducted to understand conventions and techniques that are needed for the development of a new knowledge-based system for the design of skyscrapers. I chose projects from two leading skyscraper designers, Cesar Pelli and Norman Foster. Cesar Pelli's name has long been associated with the innovative use of curtain wall systems. Norman Foster is renowned in his innovation in design technology and many of his projects show the intensive use of structural skin systems. To present a more comprehensive view of skyscraper design and construction, this section of case studies examines the non-structural systems by Cesar Pelli and the structural systems by Norman Foster.

3.1 Non-Structural Skin Projects

3.1.1 Impact of Cesar Pelli in Skyscraper Design

Since his first tower—the Museum of Modern Art Gallery Expansion in New York-- was erected in 1977, Cesar Pelli has continued to set his pace in designing high quality skyscrapers. In the past 29 years, he and his partners and associates have designed and built more than 40 skyscrapers. These skyscrapers occupy more than 75 million square feet of total floor areas. Among these built skyscrapers, Petronas Towers in Kuala Lumpur and Hong Kong International Finance Center in Hong Kong were ranked the tallest towers in the world respectively at the time they were constructed. According to Joseph Glovan-

nini, a journalist and critic in architecture, “Cesar Pelli builds the new equivalent of one-and-a-half Empire State Buildings every 12 months.”⁷

In addition to the extraordinary number of built skyscrapers, Cesar Pelli is also well-known for his “conceptual innovations” in building envelopes. For Pelli, the most important element that influences his design of a curtain wall is the “contextual response” to the surrounding environment. The skyscrapers that he has designed merge into their contexts and generate a harmony between the skyscraper and the city. The forms, materials, and details of skyscrapers are very carefully designed and crafted to reinforce the connection between the building and its local context and to strengthen the local culture.⁸

In short, the impact of Cesar Pelli in skyscraper design is two-fold. The first is in his innovative approaches on curtain wall systems that re-enforced local culture. The second lies in his emphasis on contextual response of skyscrapers to their surrounding environment. These two major contributions have been the trade-mark of Cesar Pelli in skyscraper design and construction.

7. Joseph Giovannini, *Fast Forward for Section Through A Practice: Cesar Pelli & Associates* (Editor: Raul A. Barreneche), 2005. pp56-57

8. *Ibid.*, 57

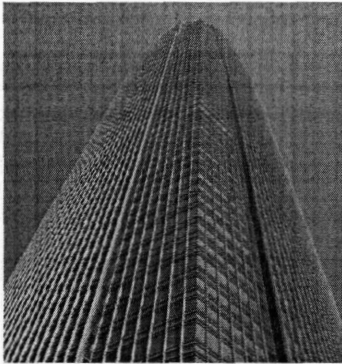


Figure 23: HKIFC by Cesar Pelli
Image shows a strong vertical impression from the carefully crafted vertical mullions. - image taken from Michael J. Crosbie's *Curtain Walls*

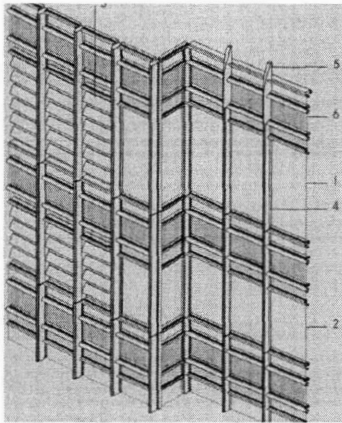


Figure 24: HKIFC by Cesar Pelli
Image shows curtain wall assemblies with mechanical louver and corner conditions - image taken from Michael J. Crosbie's *Curtain Walls*

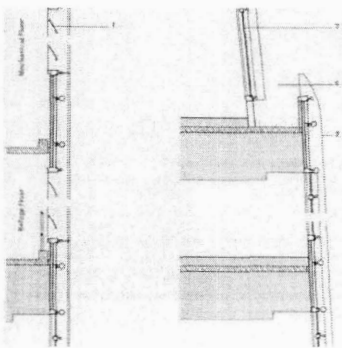


Figure 25: HKIFC by Cesar Pelli
Image shows sections on mechanical level and transitional level - image taken from Michael J. Crosbie's *Curtain Walls*

3.1.2 Project One:

Hong Kong International Finance Center

The Hong Kong International Finance Center (Figure 23) represents the third generation of Cesar Pelli's skyscraper design, which is simple in form but rich in details. (The first generation projects are read as glass boxes and the second generation projects, contrastingly, reveals the constructs of sophisticated details on facades.) The massing of the tower reflects a typical modernistic approach, yet the richness of its details detaches the tower from the typical modern style. The profile of the vertical mullions is dedicatedly crafted and sized to reinforce the verticality of the building. Transparent and translucent glass panels are carefully combined to allow better reading of the colors of the building facades for views from inside the tower. The roof is formed by four sleeves of curved walls, shaping the tower's crown when it reaches the sky.

Characterized by verticality formed by vertical mullions, the Hong Kong International Finance Center reflects the idea of reading the tower as a single entity: the designer emphasizes its vertical height over its horizontal span. Vertical mullions are made of painted aluminum with an outwards projection of 300mm. From a frontal view, these vertical mullions give us a "knife-edge" impression and describe the tower as a "glassy pylon." Seeing it from an oblique angle, the depth of the mullions is read, reducing the massiveness of this "glassy wall." (Figure 24) At the tower's indented corners, the mullions

are slender, displacing the indented corner a “glassier” appearance. Rising up to the top, these vertical mullions catch light as they go. They become smaller and smaller and eventually penetrate through the roof to form the exposed tilted wall.⁹

3.1.3 Project Two: Petronas Tower

The Petronas Tower in Kuala Lumpur, Malaysia (Figure 26) was the tallest tower in the world; its facades, on the other hand, emphasize their horizontal composition. The horizontality is achieved by re-enforcing horizontal mullions, made of either metal or stone, and by the intensive use of horizontal louvers. Also, stainless steel spandrel panels are laid out horizontally in order to maintain their consistent appearance around the entire building. The core reason behind the horizontal re-enforcement is to emphasize the identity of each individual unit that spans one column to the other. According to Edward Donnie, a project architect of Petronas Tower:

We choose to and we have to give individual attentions to each unit because all individuals, occupying any unit in the building, for this particular project, are equally important. There is no hierarchy in terms of spatial organization like that of Hong Kong International Finance Center. Each individual occupant is just too important to be ignored.¹⁰

9. Michael J. Crosbie, “International Finance Center” in *Curtain Walls*. Birkhauser-Publishers for Architecture. Basel, Switzerland. 2006. pp49

10. From a personal interview with Edward Dionie at Cesar Pelli’s office.

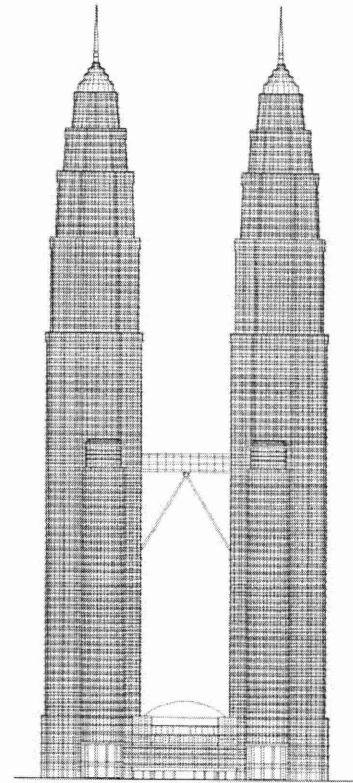


Figure 26: The Petronas Tower by Cesar Pelli
Image shows the emphasis on the horizontality of the tower. - image taken from Michael J. Crosbie’s *Curtain Walls*



Figure 27: The Petronas Tower by Cesar Pelli
Image shows the emphasis on the horizontality of the tower. - image taken from Michael J. Crosbie’s *Curtain Walls*

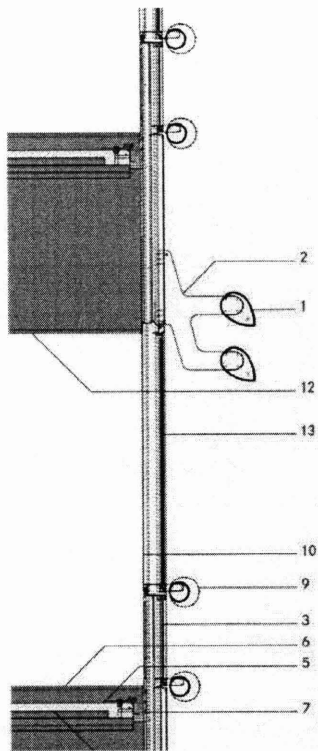


Figure 28: The Petronas Tower by Cesar Pelli
 Image shows typical section detail of the curtain wall. - image taken from Michael J. Crosbie's *Curtain Walls*

Another reason for the re-enforcement of horizontality is that it helps to reduce the massive impression of the tower to the viewers, especially for passers by on the ground level. It also changes how people read the tower. The viewers' attention is first directed to the interior units between horizontal mullions or louvers, not to the entire building (Figure 27).

A hint of a post-modernistic approach can be found in the Petronas. The details of its windows, mullions, and sun-shadings are very rich (Figure 28). However, unlike typical post-modern buildings, for which details are constructed for purely conceptual meanings, the details on the facades of the Petronas Towers are grounded under modernistic reasoning. In his new published book *Curtain Walls*, Michael J. Crosbie says that

windows are composed of continuous horizontal ribbons of modest height and protected from the sun by projecting shades. The sunshades make a three-dimensional wall, which together with the facets of the plan, create buildings with much needed shade, shadow, and dappled light; they are tropical walls. Malaysian colors, patterns, traditions, and crafts have been incorporated throughout the buildings.¹¹

11. Michael J., Crosbie, "International Finance Center" in *Curtain Walls*. Birkhauser-Publishers for Architecture. Basel, Switzerland. 2006. pp147

3.1.4 Project Three: Cira Tower

The Cira Tower in Philadelphia (Figure 29), a project that has recently been built, presents a new style for Cesar Pelli. Unlike his earlier projects that shows a clear distinction between base, body, and roof, the Cira Tower blurs this distinction with a composition of faceted surfaces in its facades. These faceted surfaces offer full views of the building from several different directions. This design approach reveals the idea of “reading the building as a sculptural monolith on the skyline.”¹² In a sense, the Cira Tower retrieves the modernistic idea of simplicity, but interprets it in a different way.

The focus of the curtain wall design supports this formal design intention (Figure 30). The mullions are suppressed and only read as very thin lines behind the glass. They reveal a two-dimensional grid, reducing their presence in the tower. The glass that is used provides a clear reading of multifaceted surfaces of the tower. Semi-reflective glass is used on both the vision and spandrel panels. The cover of the same materials throughout entire building’s surfaces effectively reduces the distinction between vision and spandrel panels.

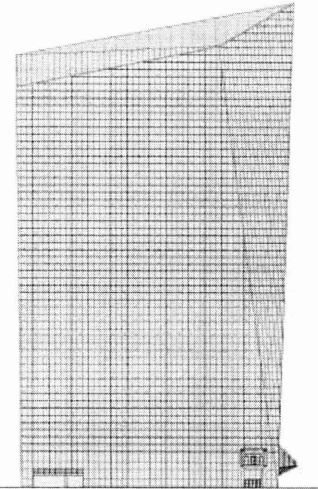


Figure 29: The Cira Tower by Cesar Pelli
Image shows multi-faceted glass box.
- image taken from Michael J. Crosbie's
Curtain Walls

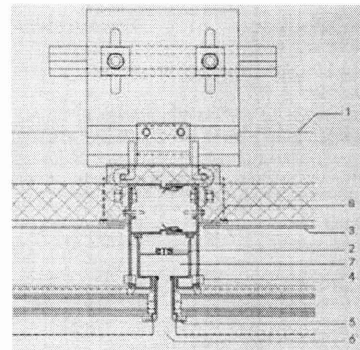


Figure 30: The Cira Tower by Cesar Pelli
Image shows typical plan detail of the
curtain wall. - image taken from Michael
J. Crosbie's Curtain Walls

¹² Michael J. Crosbie, “International Finance Center” in *Curtain Walls*. Birkhauser-Publishers for Architecture. Basel, Switzerland. 2006. pp79.

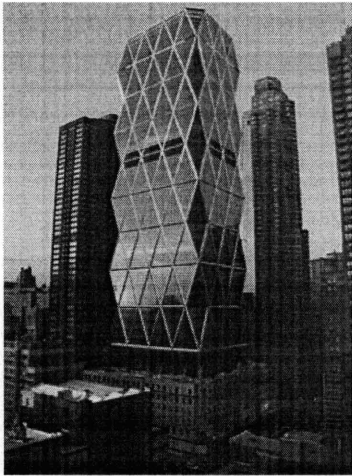


Figure 31: The Hearst Corporation Image shows structural diagrids on the facades. - image taken from www.fosterandpartners.com

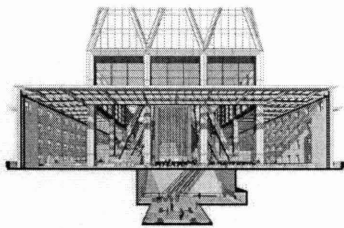


Figure 32: The Hearst Corporation Image shows the transition from structural diagrids to conventional column grids. - image taken from www.fosterandpartners.com

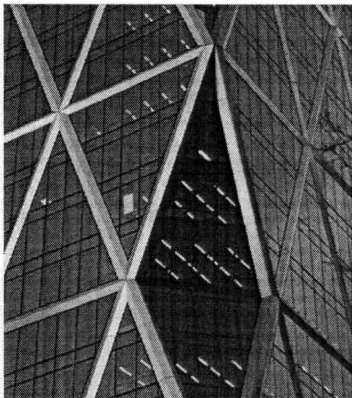


Figure 33: The Hearst Corporation Image shows a close-up look at the corner of the tower. - image taken from www.fosterandpartners.com

3.2 Structural Skin Project

3.2.1 Norman Foster and Structural Diagrids

Structural diagrids, a structural system applied on the facades of towers, was not invented by Norman Foster; its use can be dated back to the 1950s and seen in some built and un-built design projects by Fazlur Khan, Louis Khan and many others. However, it is the name of Norman Foster that is strongly associated with the use of structural diagrids in the design of skyscrapers today. Many currently built skyscrapers by Norman Forster reveal his intelligent use of this structural system on the facades of skyscrapers. Among these projects, the Swiss Re and the Hearst Corporation are the most compelling.

3.2.2 Project Four: Hearst Corporation

The Hearst Corporation tower (Figure 31) is a forty-six-story tower above the preserved façade of the old six-story Hearst Headquarter building at 959 Eighth Avenue in New York. According to Norman Foster:

The Hearst Tower expresses its own time with distinction, yet respects and strengthens the existing historical structure. The tower is lifted clear of its historic base, linked on the outside only by columns and glazing, which are set back from the edges of the site. The transparent connection floods

13. Gordon Wright, "Building on Tradition" in *Building Design and Construction*. 2005. URL: <http://www.bdenetwork.com/article/CA6161495.html>.

the spaces below with natural light and encourages the impression of the new floating above the old.¹³ (Figure 32)

The main features of the tower are the structural diagrids on its facades (Figure 33). Each of these diagrids spans twelve meters horizontally and eight-floors vertically. They not only hold the curtain walls, but also structurally support the floor slabs behind. In a sense, these triangle-shaped diagrids function both as columns and beams on the periphery of the tower. From the structural point of view, these diagrids not only satisfy structural and architectural requirements, but also allow the tower to achieve higher structural efficiency with a triangle composition. Having a long span, these diagrids eliminate the normal grids of columns in the interior of the tower and make the interior spaces much flexible for multipurpose use. These long span diagrids also allow unobstructed view from the inside of the tower to the vibrant urban setting in New York.

Chapter 4: Computational Approaches

After figuring out the geometrical rules that were extracted from the analysis of the selected projects, I develop computational methods to convert these rules into digital data, which I then embed in the system. This digital data is re-usable and modifiable to allow new design. This chapter provides detailed information about how to convert these geometrical rules into digital data and demonstrates the techniques of using these rules to construct the massing and detail of selected skyscrapers and their derivative types. Also included in this chapter is the technique of making an interface for user interaction with the program.

4.1 Structure of the Program

The runtime program is written in the Rhinoscript environment. Its functions are organized in a hierarchical tree order (Figure 34), with the highest level of function is the Main Function that asks for user selection of four points. The Main Function is also the mechanism linking the rest of the program with the user input from the interface. At the second level of this hierarchical tree is the Massing Construction Function (a function to generate the overall massing of the selected type) and the Detail Construction Function (a function to generate selected detail component).

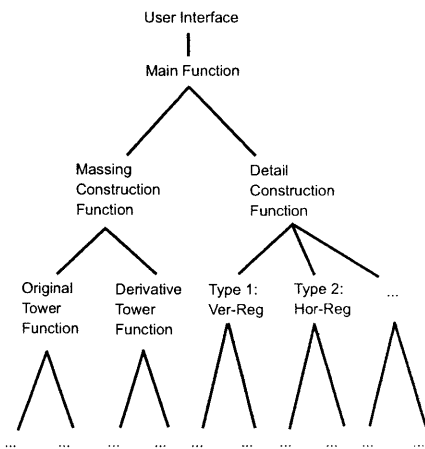


Figure 34: Tree Structure of the Program
The diagram shows the hierarchical tree structure of the program.

The Massing Construction Function is diverted into two sub functions, the Original Tower Function (a function to construct the selected original project and its pro-

portional variations) and the Derivative Tower Function (a function to construct derivative types or new types of skyscrapers from the selected original projects). Under the Original Tower Function are lists of sub-functions that actually construct each selected projects (Figure 35). These sub-functions include HKIFC Massing Function, Petronas Massing Function, Cira Massing Function, and Hearst Headquarter Massing Function. Under each sub-function is a group of functions that construct plan profiles, such as HKIFC_Plan_Profile_1 for constructing a plan profile of the Hong Kong International Finance Center. They are in the lowest level of the hierarchical tree structure. Similar tree structure composes the sub-functions of Derivative Tower Function.

The sub-tree structure of the Detail Construction Function is very similar to the Massing Construction Function. Refer to Figure 35 for details.

One advantage of having a hierarchical tree structure is that the overall frame work of the program can be set up before actual functions are written. Also, functions can be continuously sub-divided into sub-functions so that both the function and its sub-function can be reusable by all other functions at all hierarchical levels. As a result, the subdivision of functions reduces the redundancy of the coding that can be used more than once.

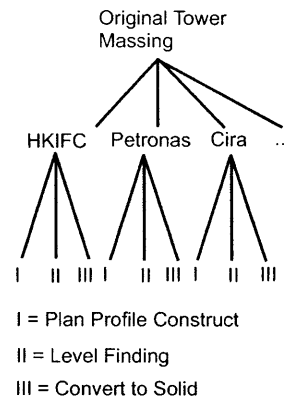


Figure 35: Sub-Tree Structure of the Proposed Program
 The diagram shows the hierarchical sub-tree structure of the program on constructing original tower massing.

4.2 Algorithmic Methods for Constructing Massing

Two computational methods are developed to construct the massing of selected projects: the Profiles Finding Method (a method that finds all profiles on all transitional levels) and the Points Finding Method (a method that finds points in space). The Profiles Finding Method uses solid modeling technique and intends for projects that have distinguished transition levels. Such projects include the Hong Kong International Finance Center and the Petronas Tower. The Points Finding Method employs surface modeling techniques and is proposed for projects that have no distinguished transition levels. The Cira Tower, a tower that is viewed as a faceted solid glass box, is constructed by using the points finding method.

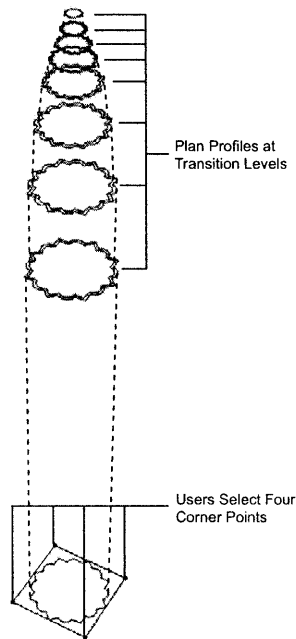


Figure 36: Profiles Finding Method
The diagram shows the construction of plan profiles on transitional levels.

4.2.1 Profile Finding Method

The Petronas Tower is used as an example for the demonstration of this technique. As the name of the method implies, the Profile Finding Method (Figure 36) constructs the plan profiles at all transitional levels; however, because these profiles are similar, 24-sided shapes, only one parametric shape needs to be defined in the system. Then, by giving the level of each profile and transformation rules, the program will construct the right profile on the assigned level. After the construction of all these profiles, the program will then loft them together to make a solid model.

The fundamental issues for the Profile Finding Method are three-fold. The first issue is to construct one typical two dimensional plan profile. The second issue is to find the level of transitions on which these profiles are located. The last issue is to find the transformation rules (scaling and rotating rules) for transforming a typical profile into the right profile on the level they are constructed.

Typical Plan Profile Construction:

The technique that I have developed for constructing a typical plan profile requires the finding of all corner points as well as all middle points on all curves (Figure 37). This technique sounds cumbersome, yet, it turns out to be the most efficient and effective method compared to many other techniques, like that of drawing lines in the first place.

Recalling the way that users interact with the program mentioned in the previous section, users only picks four corner points that define the boundary of the tower. All other points have to be set up by building the associated relationship to the chosen four corner points. Different levels of associations have been developed in order to find all the points. In the first level of association, pt1, pt2, pt3, and pt4 can be defined by their proportional relation to point1, point2, point3, and point4. (Figure 37) It is the same for pt5, pt6, pt7, and pt8. The second level of association is that I associate corner points on transitional levels to those on the first level, on which the users pick

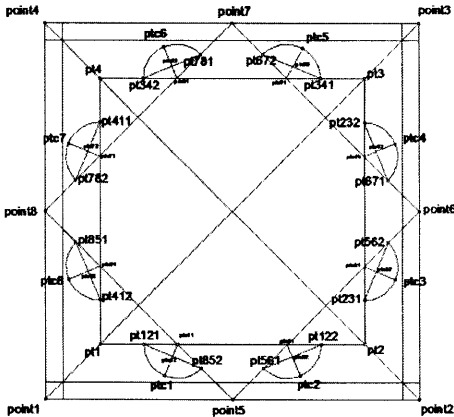


Figure 37: Typical Plan Profile
 The program constructs plan profiles by user input of four corner points and the positioning points of the profiles were preset in the system.

four corner points. As a result, we can find all the corner points. Lastly, I use the second level associated points to define the middle points on the curves. (Figure 37) After getting all these points, I use the AddLine and AddArc3Pt methods to construct the enclosed plan profile in a clockwise sequence, starting from pt1. (refer to Appendix B for a reference of the actual coding)

Level Finding

For skyscrapers in particular, there are very restricted requirements on the number of egress or refugee floors. Also, to supply the entire tower, mechanical rooms are required on multiple levels. Very typically, mechanical and refugee floors are placed together, one on top of the other. For the Petronas Tower, we can find 5 coupled levels contributing to mechanical rooms and emergency egress. It is also true that because the tower has multi-functional programs, such as retails, offices, and entertaining centers, floor heights of each level, into which functional programs are fit, differ significantly. How to reflect all these constrains in the coding present a challenge.

My solution for this challenge is to store every single required floor height into a two-dimensional Array. The sample of the code is presented here:

```
Array(5500,5500,5500,5500,5500,4100,4100,4100,4100,  
      4100,4100,4100,4100,4100,4100,4100,4100,4100,  
      4100,4100,4100,4100,4100,4100,4100,4100,4100... )
```

Different numbers in the list above represent different functional requirements. For instance, the office is facilitated with a floor height of 4100mm and retail has a 5500mm floor to floor high space. In such a way, the code can retain the actual functional requirements and embeds these requirements in the system so that the construction of the massing of the tower and its derivative or new types reflects the functional requirements. (see Appendix B for the LevelFinder function for detail)

Transformation Rules

The construction of the original towers does not require transformation rules. Functions in this section are only applied for the construction of the derivative types of skyscrapers. The core idea of using these transformation rules (scale and rotation) is that, by applied new rules together with the inherited features from the selected projects, new design can be discovered. The intention behind this exploration is to test how rule-based methods can transform previous design ideas into new design.

The way derivative types of skyscrapers are constructed is by using these transformation rules on the construction of the top profile and the middle profile. The bottom profile is set by the four corner points that users choose in the beginning.

In terms of scaling functions, users can control two things: the scale on the X axis and that on the Y axis. (Fig-

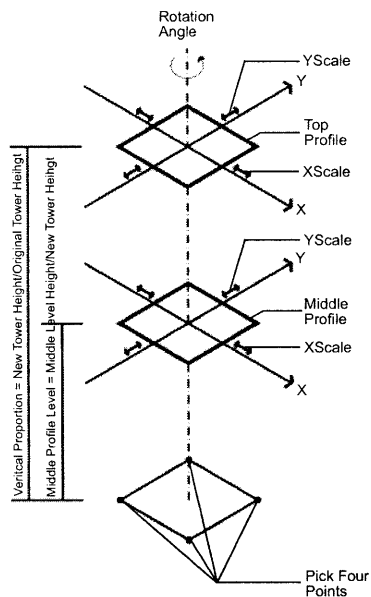


Figure 38: Transformational Rules
The diagram identifies the transformation rules used by the program.

ure 38) The range of scaling is set from 0.1 to 2 and only 0.1 increments are allowed. Over this range, towers generated through the scaling features seem unrealistic for practical reasons, as I have discovered.

The rotation features allow the rotation of the middle and top profiles to be rotated along the Z axis in a range from 0 to 180 degree in an increment of 10. (Figure 38) Once again, this setting aims to prevent some unrealistic and unpractical options to be generated.

4.2.2 Point Finding Method

As the name implies, the Point Finding Method attempts to find all points in space to construct the tower. This technique is only applied to projects that contain multi-faceted surfaces and that contain a reasonable number of points. As a demonstration of this technique, I use the Cira Tower as an example.

As I have previously described in Chapter 3, there is logical reasoning behind the construction of all these multi-faceted surfaces in the Cira Tower project. A graphical representation of the relationship among these points in space is shown on Figure 39. The method of locating these points in space is to find the associated relationship between these points and the four corner points that users select. Then, sets of lines are drawn from the connections of these points in space to define the edges of each faceted surface. This is done by using the AddLine method in

Rhino. As previously mentioned, the Point Finding Method uses the surface modeling technique. The last step is to convert sets of lines into surfaces by using the AddLoftSrf function. Then, all surfaces are joined to become a single solid entity. (Figure 39)

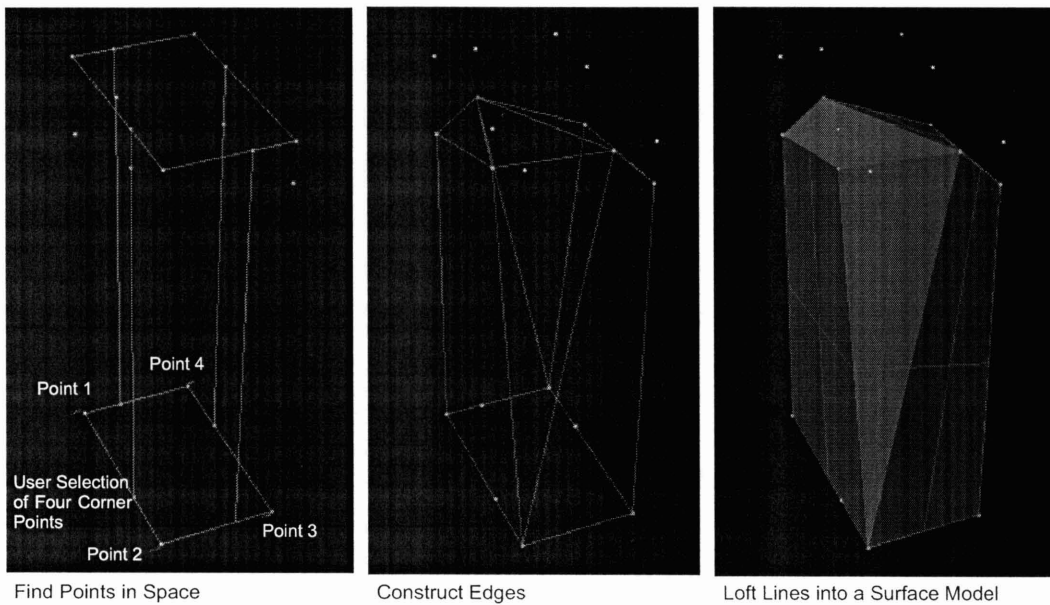


Figure 39: Massing Construction Procedure
The diagram three steps of constructing massing by using Points Finding Method:
1. find points in space; 2. connect points to find the edges of the building; 3. construct a surface model by lofting lines.

4.3 Algorithmic Methods for Constructing Detail

Three issues are considered for the construction of detailed components. The first is the positioning of the components. The second is the construction of plan and section profiles. The last one is the methods to convert two-dimensional profiles into three-dimensional solid models. In order to locate a component in a preferred position, I develop an algorithm to find the associative relationship between positioning points and their adjacent points, in which the next components are generated. In such a way, the relationship between a component and its adjacent components is set. Responding to two different types of components, structural and non-structural, two methods are developed. They are the Dividing Points Finding Method and the Diagrid Points Finding Methods.

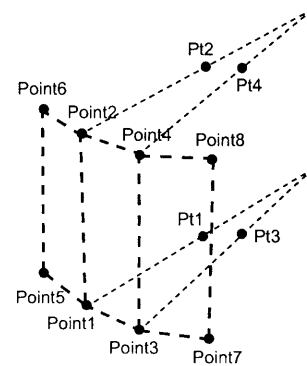
Techniques are also developed to construct three-dimensional models. These techniques include methods to generate plan and section profiles as well as that to convert profiles into three-dimensional solid models. The following section provides practical information about these algorithmic methods.

4.3.1 Associative Relationship between Positioning Points and Adjacent Points

Why do we need to figure out the associative relationship between position points and adjacent points? Let

us take the installation of a window as an example to explain. To install a window, we first find the area, in which the window is going to be installed. Usually, this area is defined by four corner points. Then, we need to decide the orientation of the window. Installing a window physically, we can find the orientation by inspecting the actual site conditions—the outside face of the window faces the outside and the window is aligned with the edge of the wall. However, in order to do the same thing in computer, we need to do it in a different way because the computer does not recognize the inside, outside, and the edge conditions. This bias between what we see and what computer sees requires an extra mechanism to help the computer identifying the actual physical conditions. By finding the associative relationship between position points and adjacent points, we can inform the computer about both the orientation and the edge condition.

Figure 40 demonstrates how this associative relationship works. Position points (Point1, Point2, Point3, and Point4) indicate the area of the component to be constructed. Adjacent points (Point5, Point6, Point7, and Point8) show the edges of adjacent components on the left and right hand sides. In such a way, a relationship between a component and its adjacent components are set. So, how can we find the orientation in this associative relationship? It can be done by finding the positions of reference points (Pt1, Pt2, Pt3, and Pt4). These reference points identify the inner area. When we construct a component, we encounter three different kinds of geo-

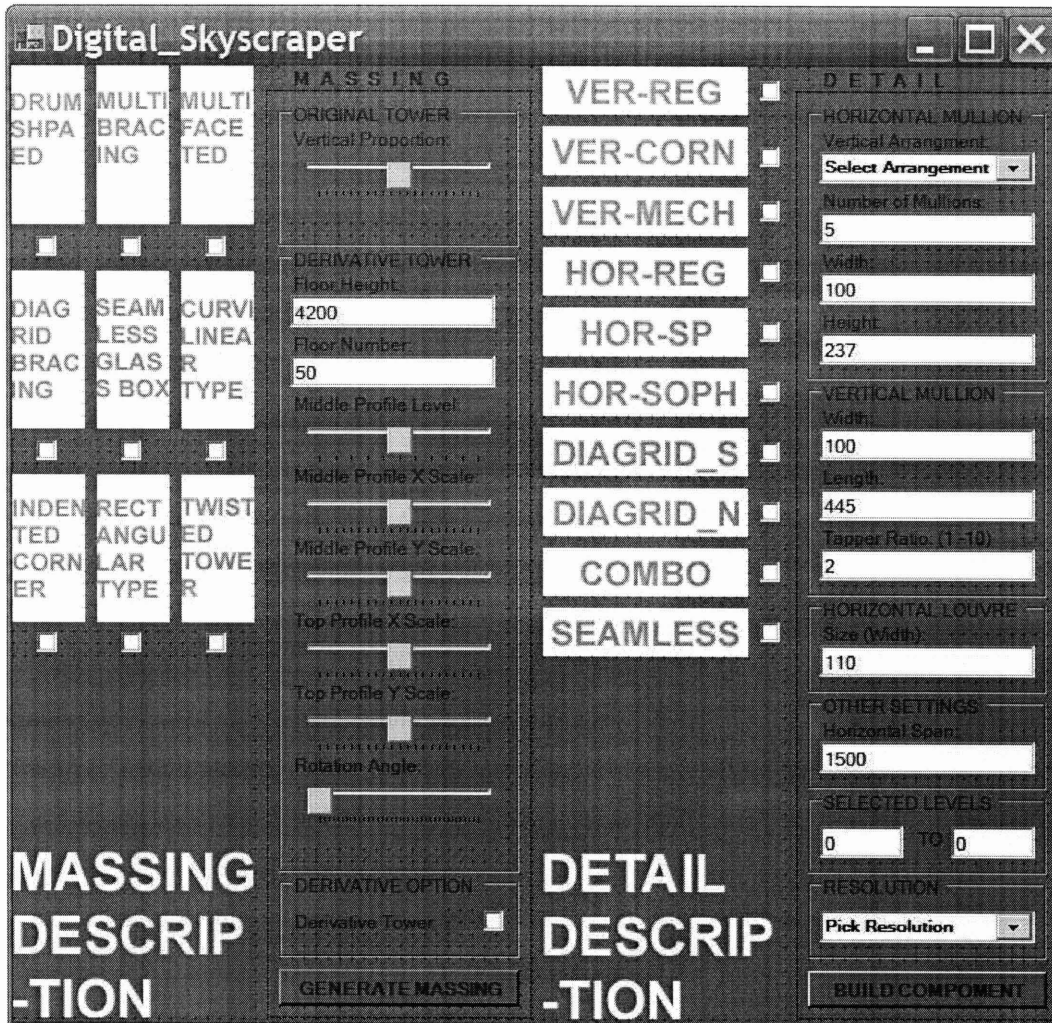


Position Points: Point1, Point2, Point3, Point4
 Adjacent Points: Point5, Point6, Point7, Point8
 Reference Points: Pt1, Pt2, Pt3, Pt4

Figure 40: Associative Relationships between Position Points and Adjacent Points. The diagram shows the way to construct a component on selected position points by setting up the relationships among position points, adjacent points, and reference points.

metrical conditions: concave, convex, and aligned conditions. Methods for finding these reference points can be found in Appendix B.

In order to construct components recursively on a selected surface, two methods are developed to handle two different kinds of components: the Dividing Points Finding Method for non-structural components and the Diagrid Points Finding Method for structural components. The Dividing Points Finding Method (see Appendix B for actual codes) takes a surface, finding the level line on it; then it divides the level line into a number of divisions that have been inputted by the user. On both sides of the edges of the surface, the ExtraPointFinding Method is used to find additional reference points. In a different way, the Diagrid Points Finding Method not only finds division points on the level line, but also finds two reference points on both sides of a generated division point. These reference points are taken as position points for generating components.



Settings for Massing:

1. Vertical Proportion: for construct original towers and its variations that have extra same composition with the original towers.
2. Massing Description: provide detail information of the type of massing user selected.
3. Floor Height: floor to floor height by user input
4. Floor Number: number of floors by user input
5. Middle Profile Level: the location of the middle profile
6. Middle Profile XScale: the scale factor on X axis of the middle profile
7. Middle Profile YScale: the scale factor on Y axis of the middle profile
8. Top Profile XScale: the scale factor on X axis of the Top profile
9. Top Profile YScale: the scale factor on Y axis of the Top profile
10. Rotation Angle: the angle of rotation of the top profile

Settings for Detail:

1. Detail Description: provide detail information of the type of detail
2. Vertical Arrangement: the vertical position of horizontal mullions and louvers
3. Horizontal Mullion Width: the width of mullion by user input
4. Horizontal Mullion Height: the height of mullion by user input
5. Middle Profile Level: the location of the middle profile
6. Vertical Mullion Width: the width of mullion by user input
7. Vertical Mullion Height: the height of mullion by user input
8. Vertical Mullion Tapper Ratio: the ratio of tapering. Actual dimension equals to number of user input * 5mm
9. Horizontal Louver: the size of the horizontal louver
10. Horizontal Span: the horizontal span of each component
11. Selected Levels: the level on which components to be constructed
12. Resolution: low, medium or high resolution for different applications

Figure 40: User Interface for User interaction with the program

4.4 User Interface

Figure 40 shows the user interface. It was written in VB.Net environment and contains two sets of parameters. The first set in the Massing Control Panel is of controlling the massing generation process. The second set in the Detail Generation Panel contributes to the detail generation.

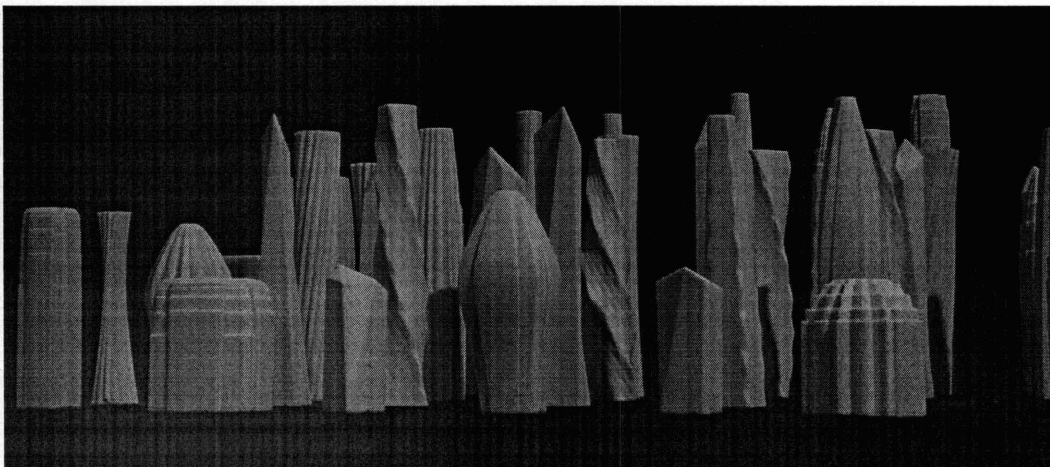
In the Massing Control Panel, two major components are laid out side by side. On the left is the descriptive component, providing the types of massing and the detailed descriptions of each type. On the right is the parameter input component, a mechanism for user inputs of parameters by ways of TextBox, TrackBar, or CheckBox. User has controls on the vertical proportion, floor height, floor number, middle and top profiles scales, and rotation angle. A Derivative Option CheckBox offers an option to either construct the original tower or its derivative types.

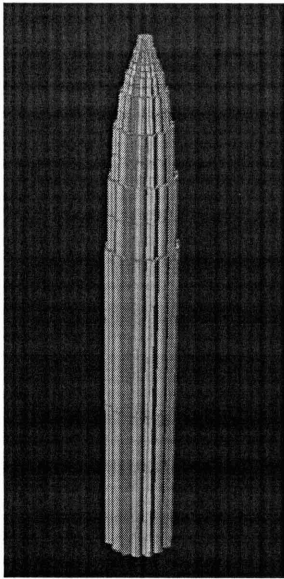
Similar to the layout of the Massing Control Panel, the Detail Generation Panel offers the selections and descriptions of detail types as well as parameters for controlling detail generation process. These parameters allow user's control on the height, width and number of horizontal and vertical mullions, the tapering ratio of the vertical mullions, and louver sizes. The Selected Levels and the Resolution Selection parameters provide options to construct details at the selected levels in a chosen resolution setting.

Chapter 5: Experiments

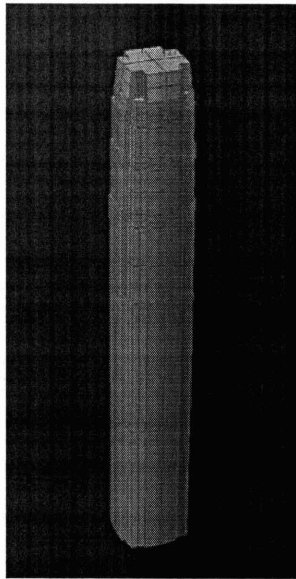
In previous chapters, in-depth case studies of selected projects were conducted and computational methods were developed to capture rules and design intentions of these projects. In this chapter, by doing experiments on reconstructing the facades of original design and their variations, I investigate the validity and the range of application of the proposed system. The first part of this experiment shows the result of constructing the massing of four original types of towers, namely 16-Branch Form, Set-back Corner Type, Multi-Faceted Glass Box, and Structural Skin Mass (Figure 41). The second part of the experiments shows the result of constructing three types of details on a surface or at selected levels of a building. Types of details are the Reinforced Verticality, the Reinforced Horizontality, and the Structural Diagrid. By incorporating the massing and details, the last part of this experiment presents two new towers: the Drum-Shaped Tower and the New Twisted Tower.

Figure 41: 3D Printed Models
The photo shows variations generated by the program.

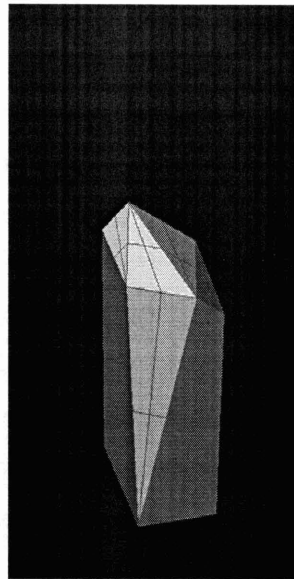




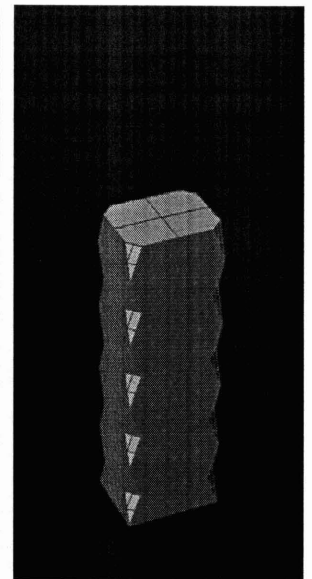
16-Branch Form
(Original Design by
Cesar Pelli)



Set-back Corner Type
(Original Design by
Cesar Pelli)

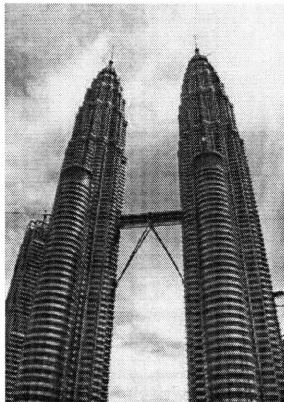


Multi-Faceted Glass Box
(Original Design by
Cesar Pelli)

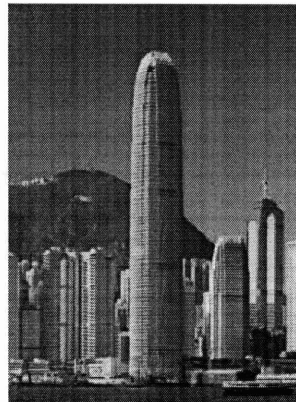


Structural Skin Mass
(Original Design by
Norman Foster)

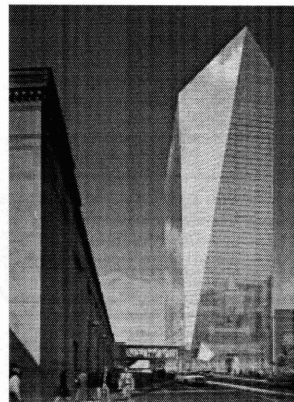
Figure 42: 3D Models of Original Design
constructed by the proposed system



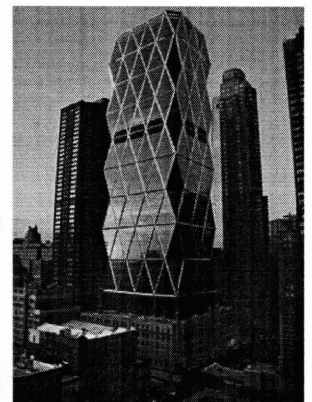
Petronas Tower
(by Cesar Pelli)



HKIFC
(by Cesar Pelli)



Cira Tower
(by Cesar Pelli)



Hearst Cooperation
(by Norman Foster)

Figure 43: Original Design Projects

5.1 Massing

5.1.1 Types of Massing

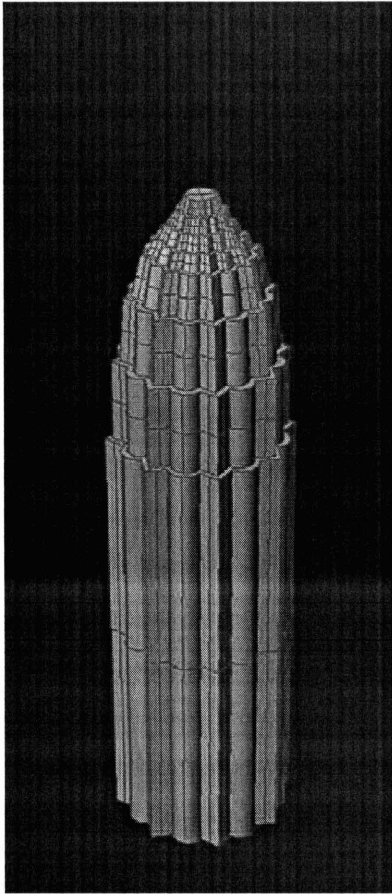
Figure 42 presents four original massing types being generated by the system: the Set-back Corner Type, the 16-Branch Form, the Multi-Faceted Glass Box, and the Structural Skin Mass. Variations are shown on subsequent pages. Also included in the subsequent pages are settings that were used to generate these towers. Three-dimensional Z-Corp models were made to review the physical outcomes of the system and demonstrate one of its potential applications: to generate digital models that are compactable to current three-dimensional printers.

16-Branch Form: is generated by using the Profiles Finding Methods. It tapers when it rises to the sky.

Set-back Corner Type: is generated by using the Profiles Finding Methods. It has indented corners and sets back at transitional floors.

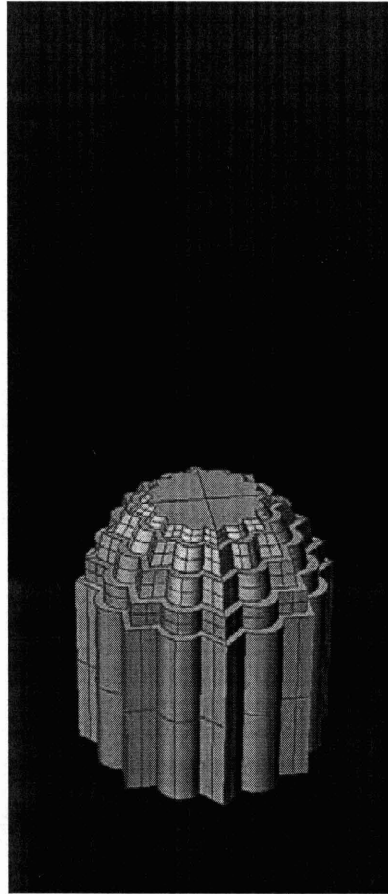
Multi-Faceted Glass Box: is generated by the Points Finding Methods. It is multi-faceted and provides dynamic view to the tower.

Structural Skin Mass: is generated by the Points Finding Methods. It presents structural diagrids on its facades.



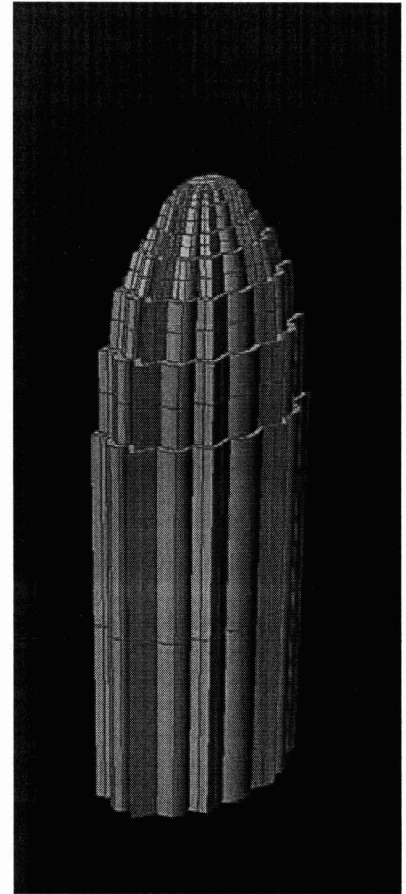
Settings:

Vertical Proportion: 0.6



Settings:

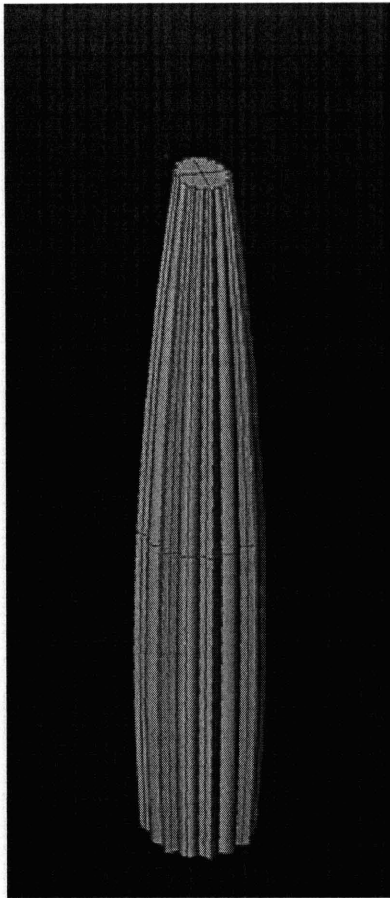
Vertical Proportion: 0.45



Settings:

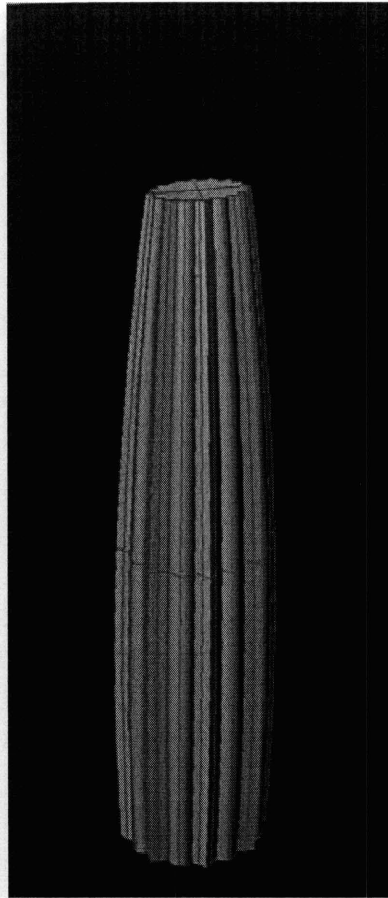
Vertical Proportion: 0.75

Figure 44: Design Variations of the Petronas Tower Generated by the Proposed System
These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanations of these rules can also be found in Chapter 4.



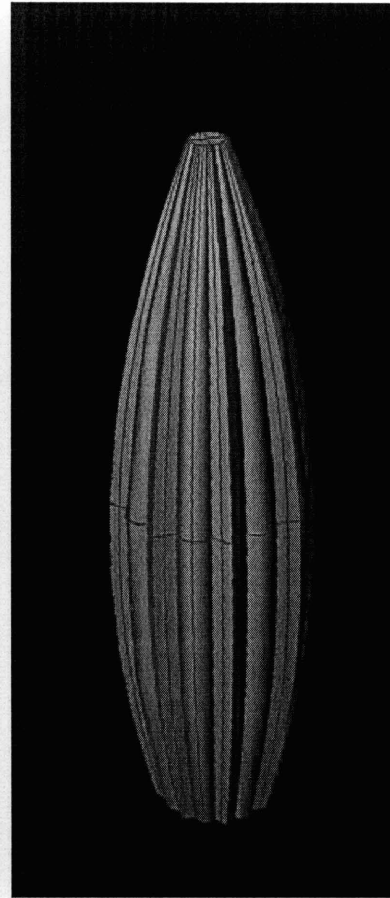
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.05
 Middle Profile YScale: 1.05
 Top Profile XScale: 0.3
 Top Profile YScale: 0.45
 Rotation Angle: 0



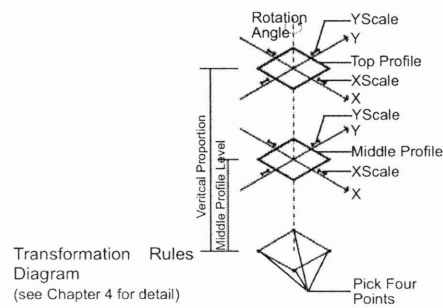
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.05
 Middle Profile YScale: 1.25
 Top Profile XScale: 0.75
 Top Profile YScale: 0.35
 Rotation Angle: 0

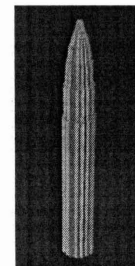


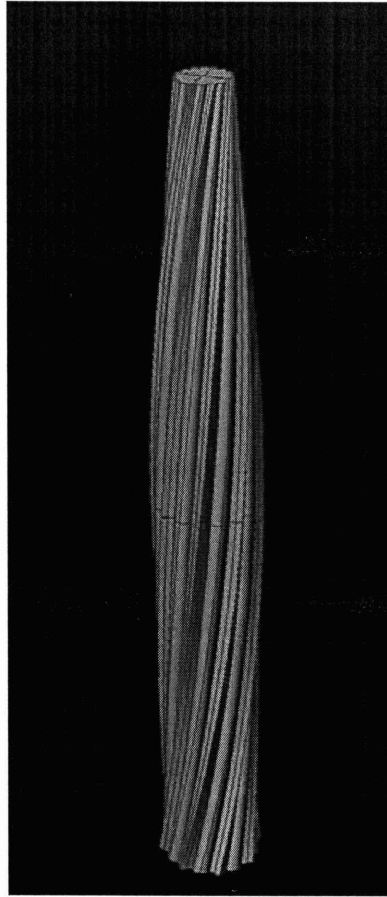
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.35
 Middle Profile YScale: 1.35
 Top Profile XScale: 0.25
 Top Profile YScale: 0.25
 Rotation Angle: 0



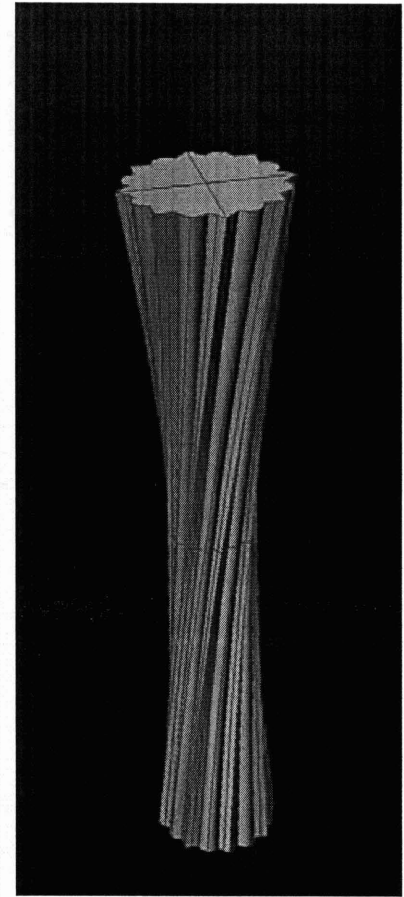
Original Type:
 16-Branch Form





Settings:

Floor Height: 4200
 Floor Number: 85
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.05
 Middle Profile YScale: 1.05
 Top Profile XScale: 0.35
 Top Profile YScale: 0.45
 Rotation Angle: 180

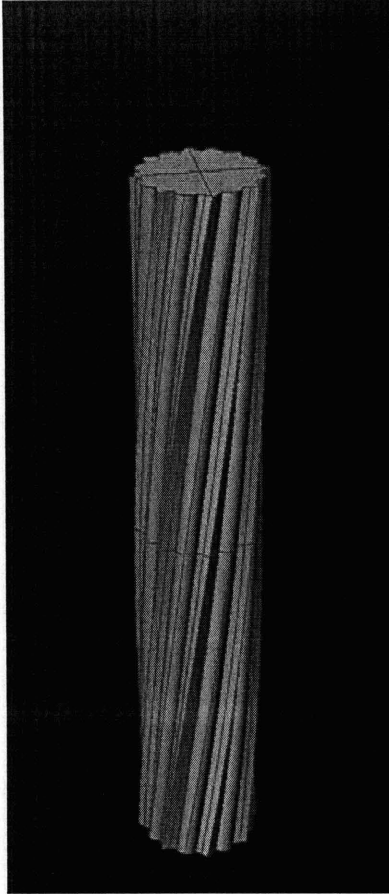


Settings:

Floor Height: 4200
 Floor Number: 70
 Middle Profile Level: 0.5
 Middle Profile XScale: 1
 Middle Profile YScale: 1
 Top Profile XScale: 1.25
 Top Profile YScale: 1.25
 Rotation Angle: 180

Figure 46: Design Variations of the Petronas Tower Generated by the Proposed System

These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanations of these rules can also be found in Chapter 4.



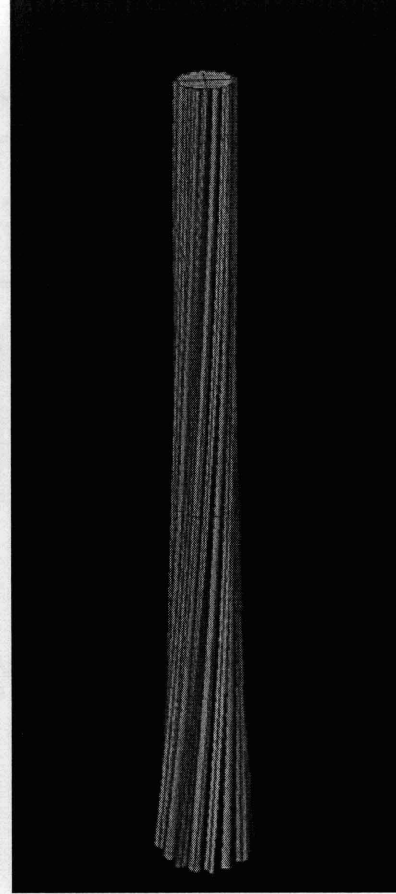
Settings:

Floor Height: 4200
 Floor Number: 70
 Middle Profile Level: 0.5
 Middle Profile XScale: 1
 Middle Profile YScale: 1
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 90



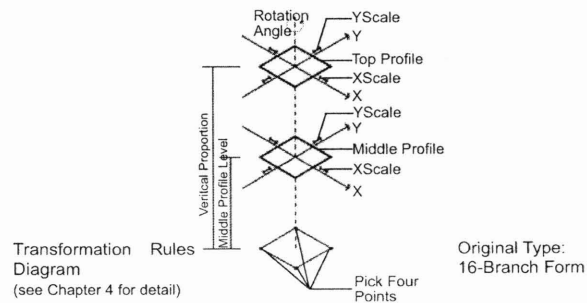
Settings:

Floor Height: 4200
 Floor Number: 55
 Middle Profile Level: 0.5
 Middle Profile XScale: 0.35
 Middle Profile YScale: 0.35
 Top Profile XScale: 0.75
 Top Profile YScale: 0.75
 Rotation Angle: 45



Settings:

Floor Height: 4200
 Floor Number: 85
 Middle Profile Level: 0.5
 Middle Profile XScale: 0.85
 Middle Profile YScale: 0.85
 Top Profile XScale: 0.65
 Top Profile YScale: 0.65
 Rotation Angle: 90



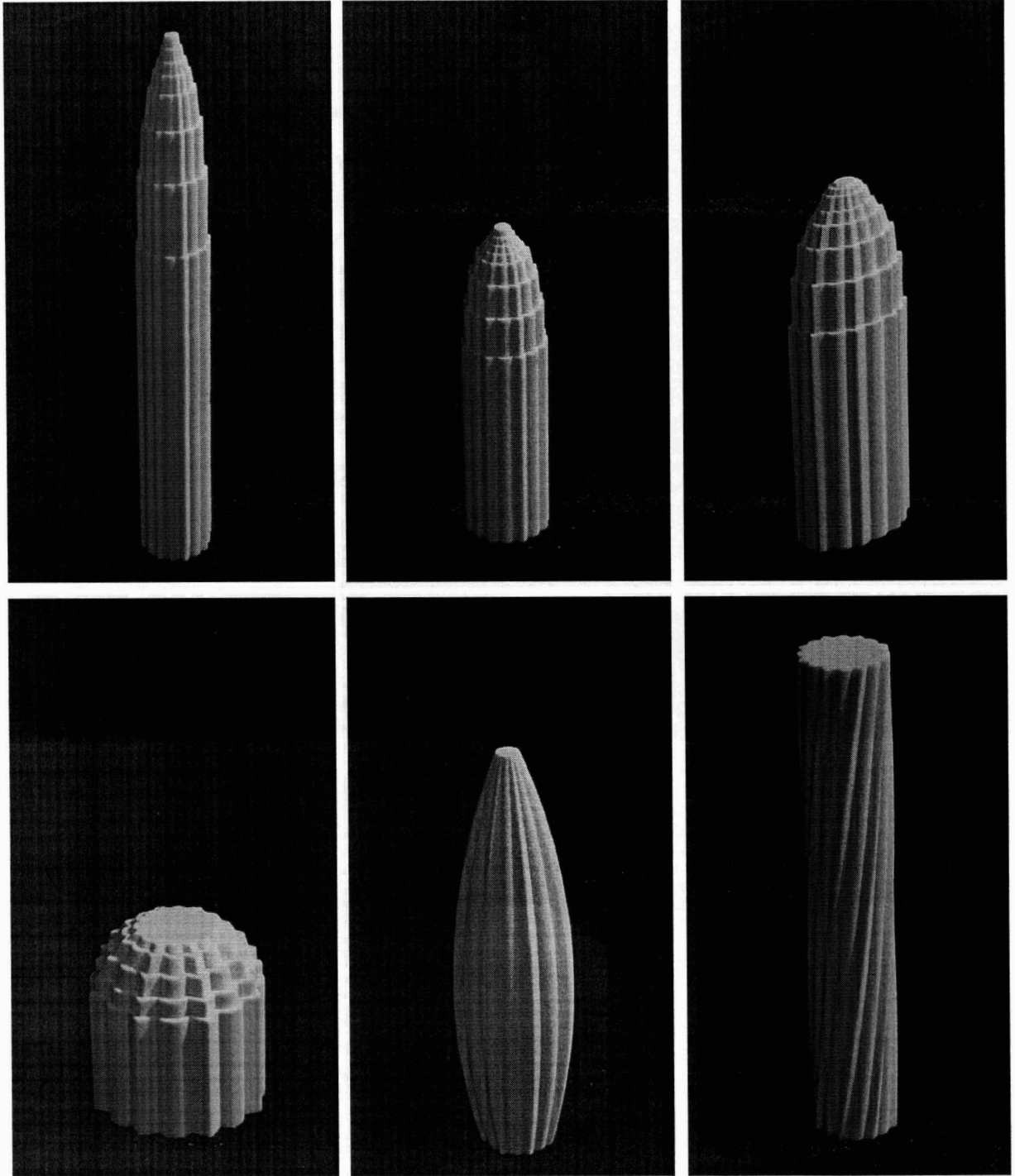


Figure 48: 3D Printed Models Showing Design Variations of the Petronas Tower

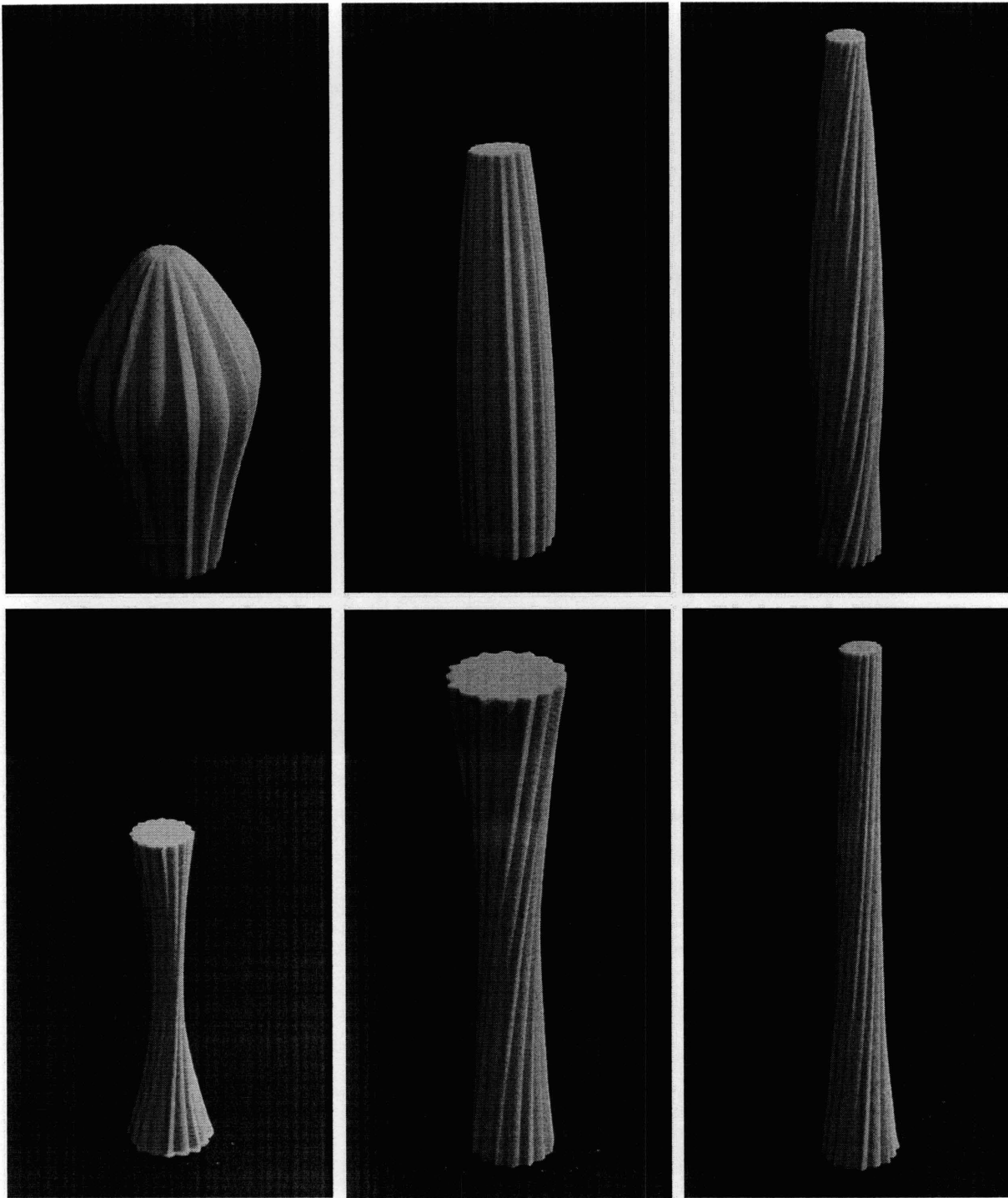
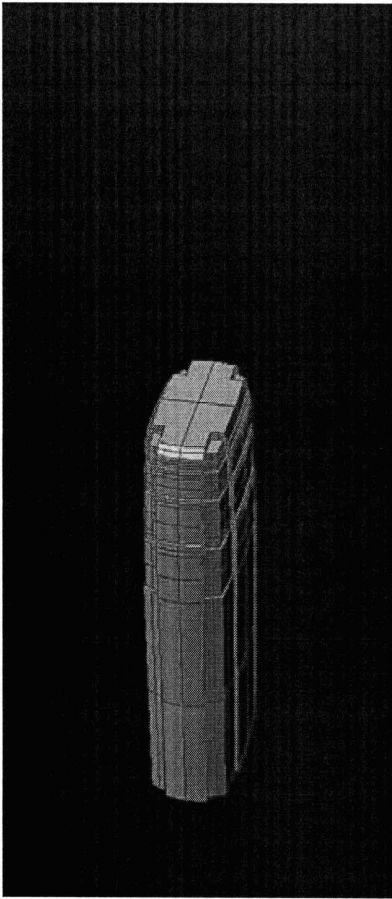
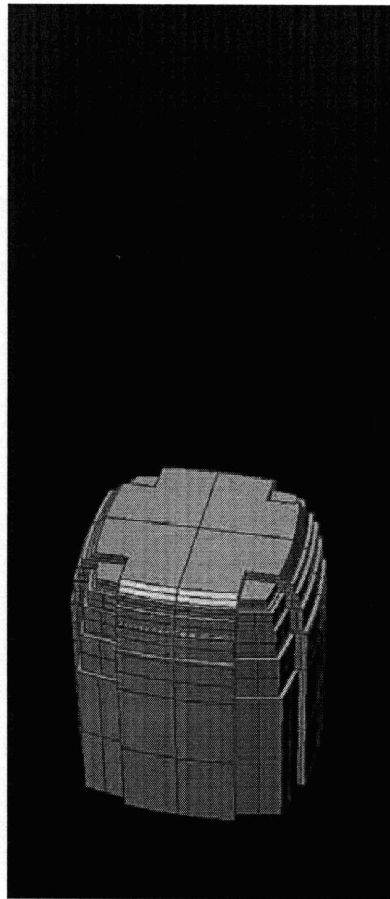


Figure 49: 3D Printed Models Showing Design Variations of the Petronas Tower



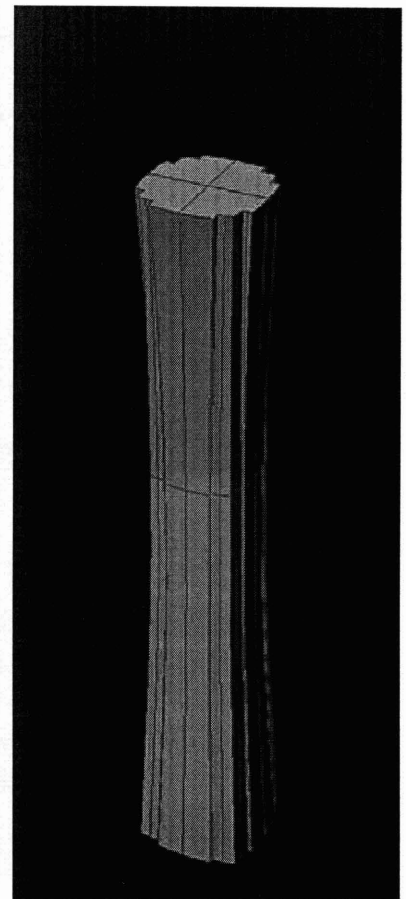
Settings:

Vertical Proportion: 0.6



Settings:

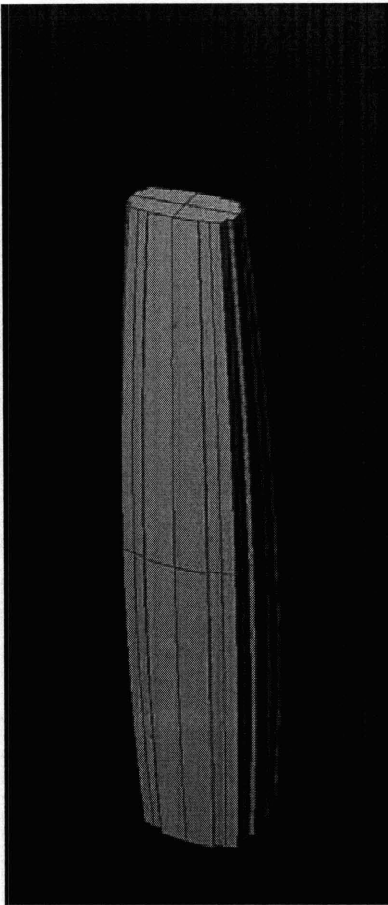
Vertical Proportion: 0.45



Settings:

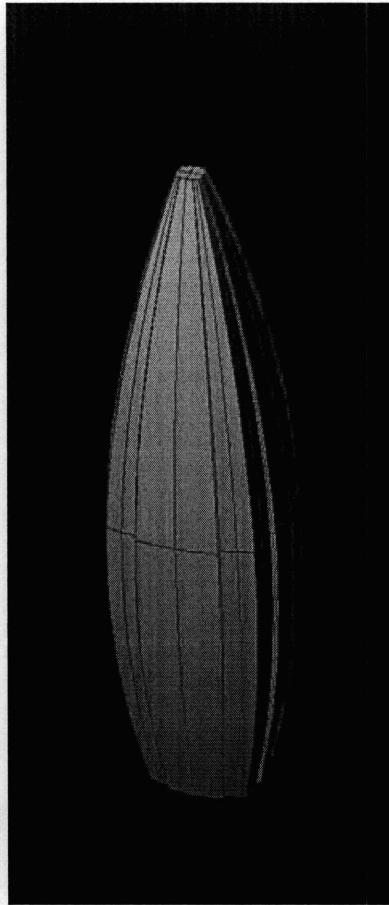
Floor Height: 4200
Floor Number: 75
Middle Profile Level: 0.5
Middle Profile XScale: 0.85
Middle Profile YScale: 0.85
Top Profile XScale: 1
Top Profile YScale: 1
Rotation Angle: 0

Figure 50: Design Variations of HKIFC
Generated by the Proposed System
These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanations of these rules can also be found in Chapter 4.



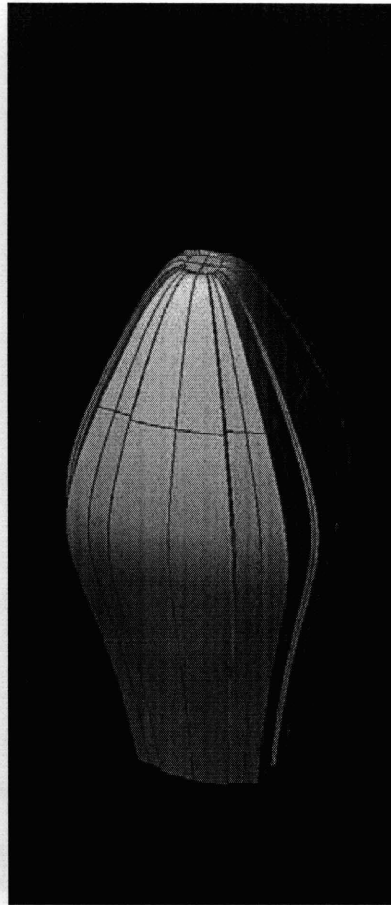
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.15
 Middle Profile YScale: 1.15
 Top Profile XScale: 1
 Top Profile YScale: 0.35
 Rotation Angle: 0



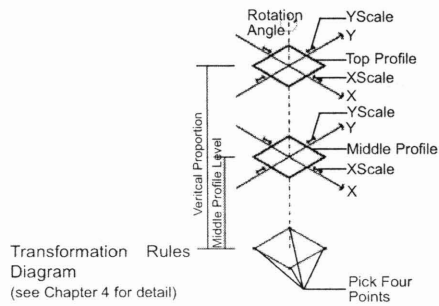
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.25
 Middle Profile YScale: 1.25
 Top Profile XScale: 0.25
 Top Profile YScale: 0.25
 Rotation Angle: 0

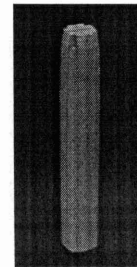


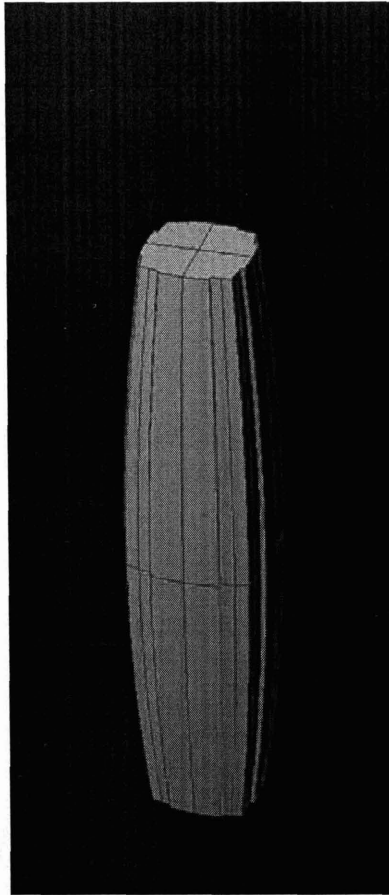
Settings:

Floor Height: 4200
 Floor Number: 50
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.45
 Middle Profile YScale: 1.45
 Top Profile XScale: 0.25
 Top Profile YScale: 0.25
 Rotation Angle: 0



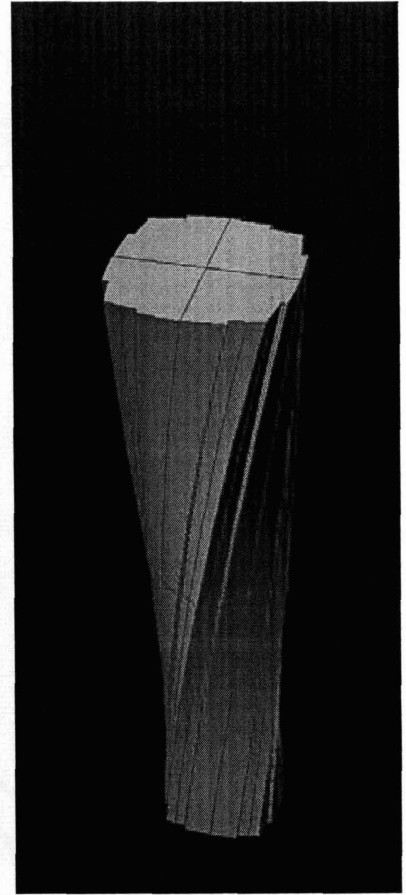
Original Type:
 Set-back Corner Type





Settings:

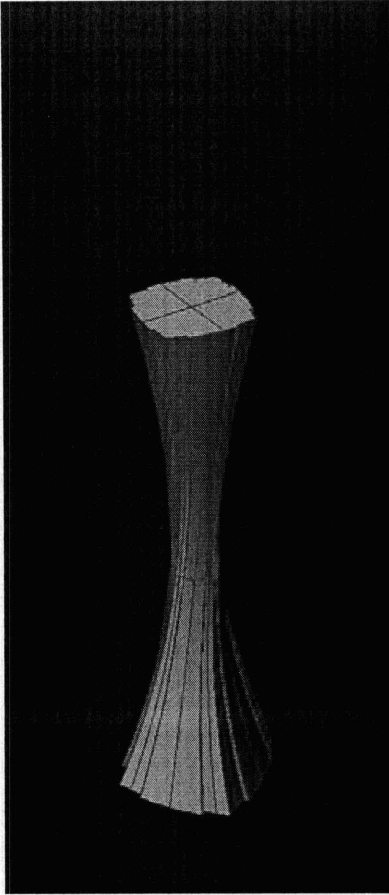
Floor Height: 4200
 Floor Number: 55
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.15
 Middle Profile YScale: 1.15
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 0



Settings:

Floor Height: 4200
 Floor Number: 70
 Middle Profile Level: 0.5
 Middle Profile XScale: 1.05
 Middle Profile YScale: 1.05
 Top Profile XScale: 1.25
 Top Profile YScale: 1.25
 Rotation Angle: 180

Figure 52: Design Variations of HKIFC
 Generated by the Proposed System
 These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanation of these rules can also be found in Chapter 4.



Settings:

Floor Height: 4200
 Floor Number: 45
 Middle Profile Level: 0.5
 Middle Profile XScale: 0.55
 Middle Profile YScale: 0.85
 Top Profile XScale: 0.85
 Top Profile YScale: 0.85
 Rotation Angle: 45



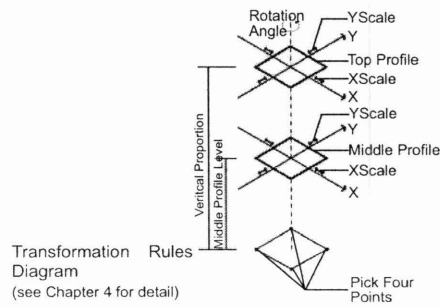
Settings:

Floor Height: 4200
 Floor Number: 85
 Middle Profile Level: 0.5
 Middle Profile XScale: 0.9
 Middle Profile YScale: 0.9
 Top Profile XScale: 0.85
 Top Profile YScale: 0.85
 Rotation Angle: 45

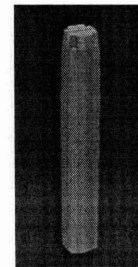


Settings:

Floor Height: 4200
 Floor Number: 70
 Middle Profile Level: 0.5
 Middle Profile XScale: 1
 Middle Profile YScale: 1
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 45



Original Type:
 Set-back Corner Type



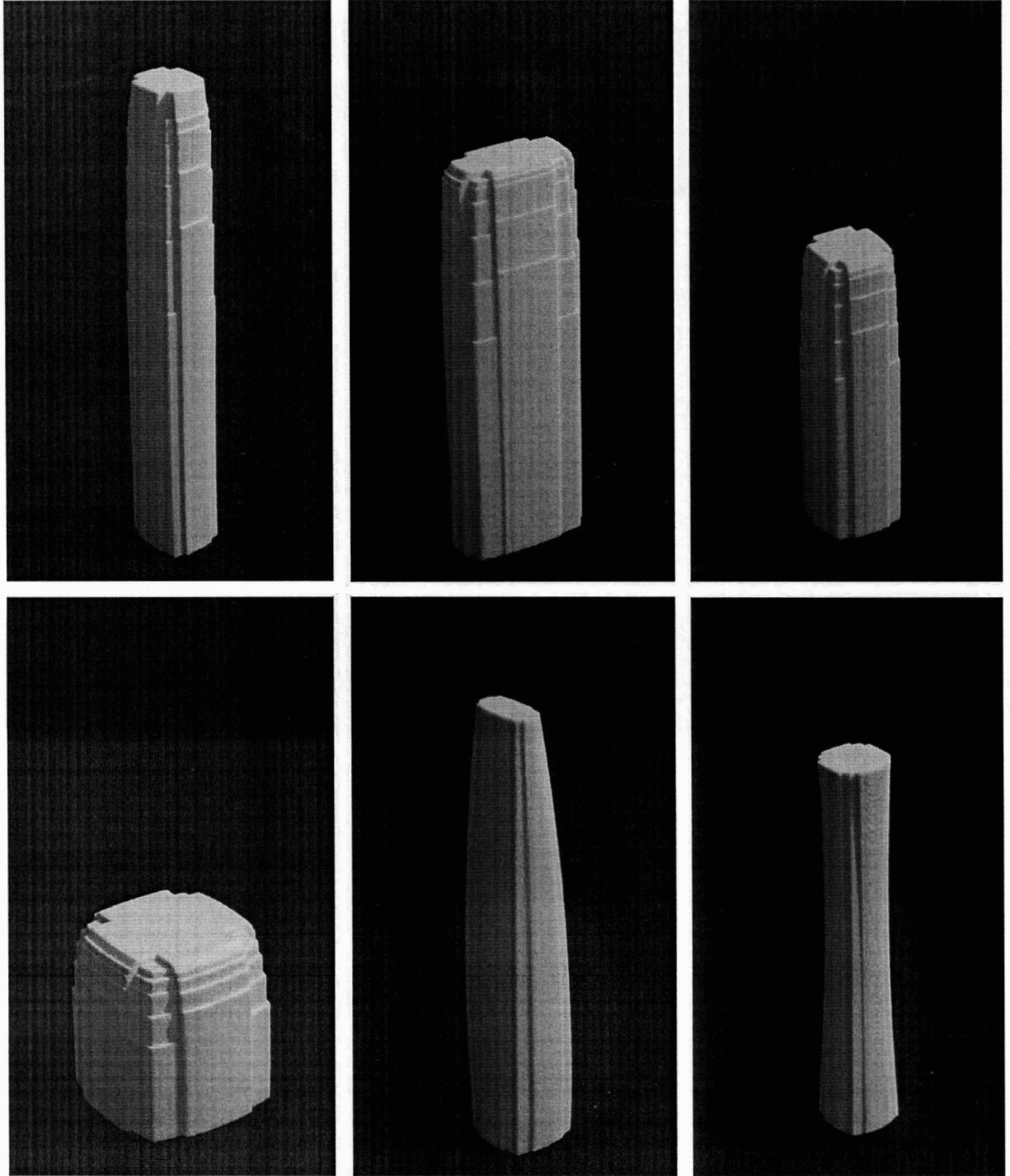


Figure 54: 3D Printed Models Showing Design Variations of HKIFC

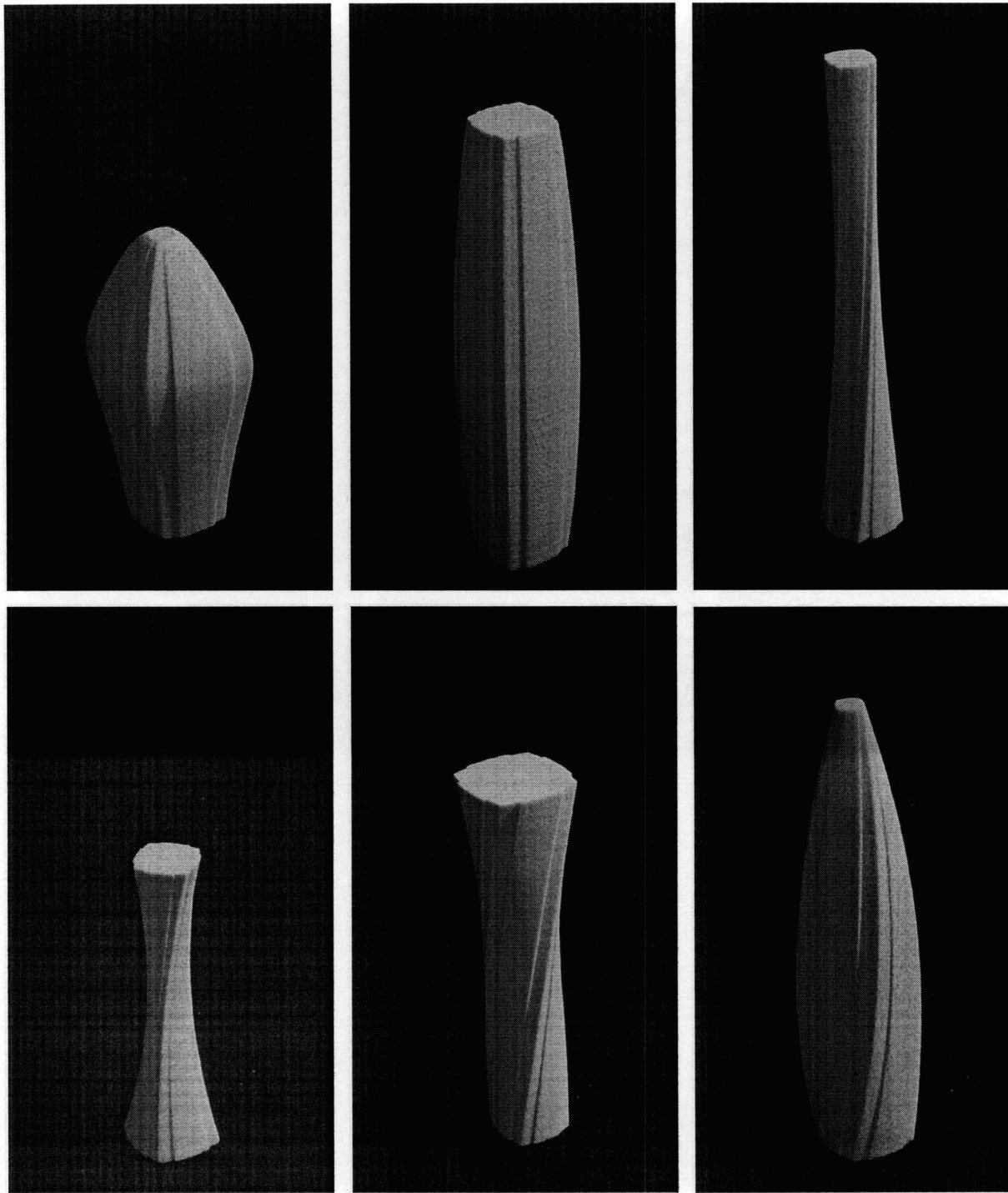
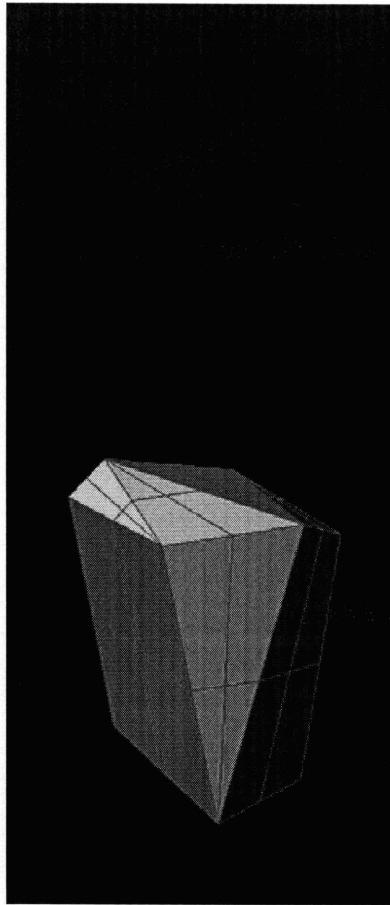


Figure 55: 3D Printed Models Showing Design Variations of HKIFC



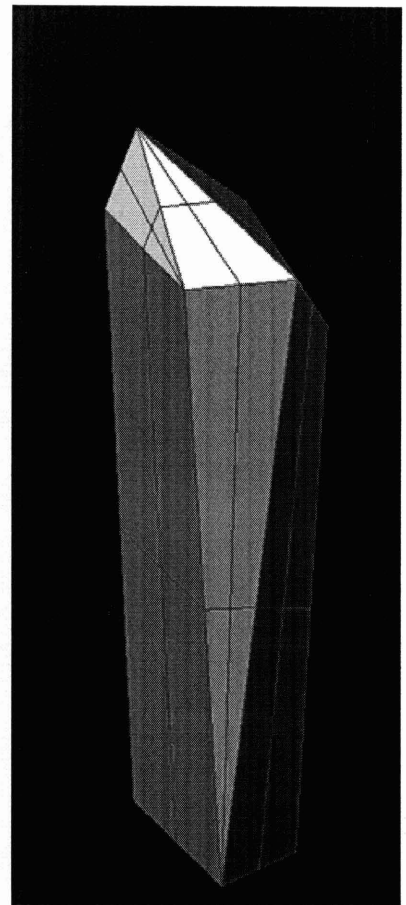
Settings:

Vertical Proportion: 2



Settings:

Floor Height: 4200
 Floor Number: 35
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1.25
 Top Profile YScale: 1.25
 Rotation Angle: 0



Settings:

Floor Height: 4200
 Floor Number: 70
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1.15
 Top Profile YScale: 1.15
 Rotation Angle: 0

Figure 56: Design Variations of the Cira Tower
 Generated by the Proposed System
 These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanations of these rules can also be found in Chapter 4.



Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.9
 Top Profile YScale: 0.9
 Rotation Angle: 0



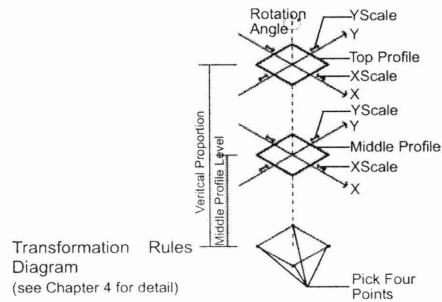
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 90

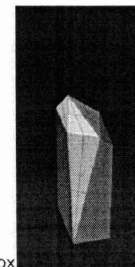


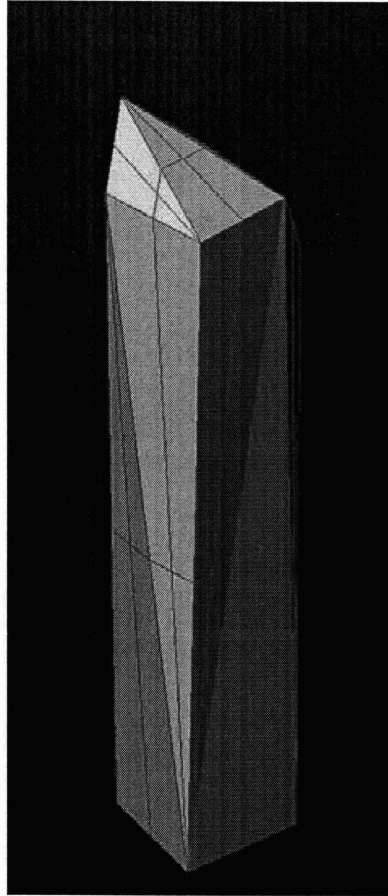
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 135



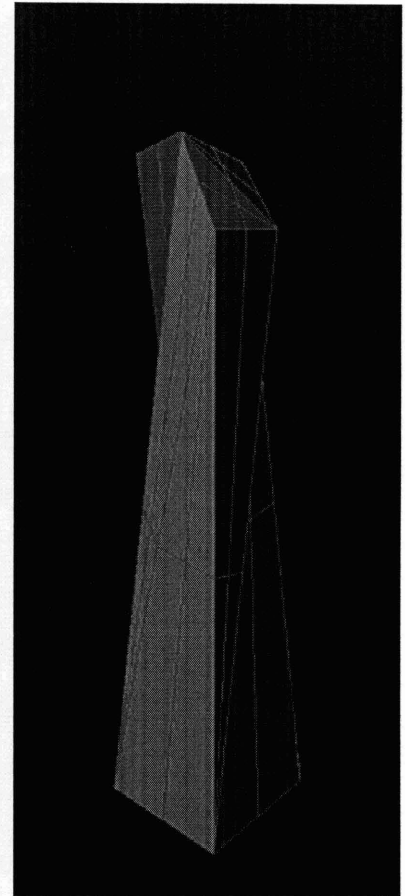
Original Type:
 Multi-Faceted Glass Box





Settings:

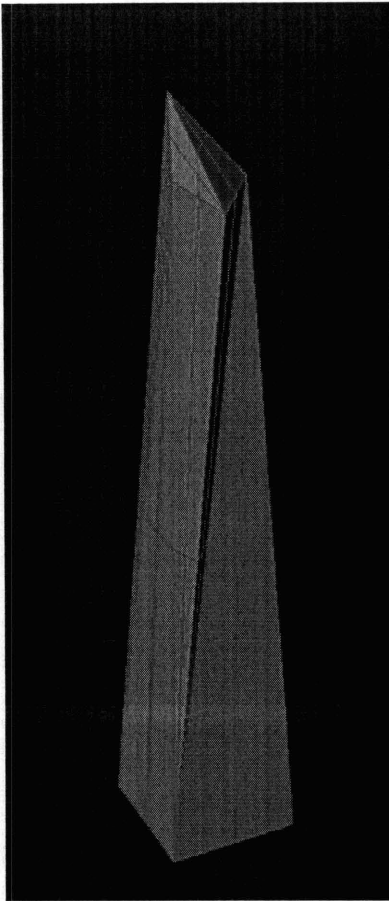
Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 90



Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.85
 Top Profile YScale: 1
 Rotation Angle: 270

Figure 58: Design Variations of the Cira Tower Generated by the Proposed System
 These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanation of these rules can also be found in Chapter 4.



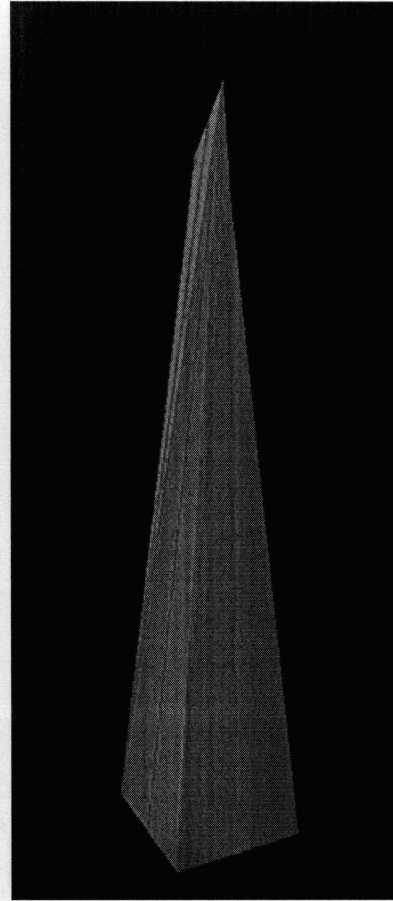
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.6
 Top Profile YScale: 0.6
 Rotation Angle: 1.35



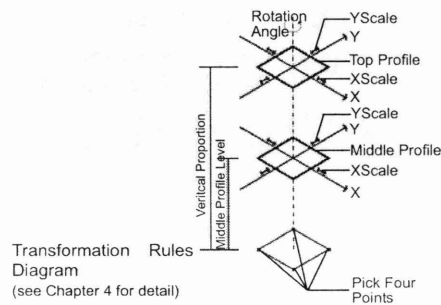
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.5
 Top Profile YScale: 0.5
 Rotation Angle: 180

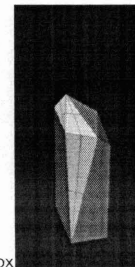


Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.25
 Top Profile YScale: 0.25
 Rotation Angle: 215



Original Type:
 Multi-Faceted Glass Box



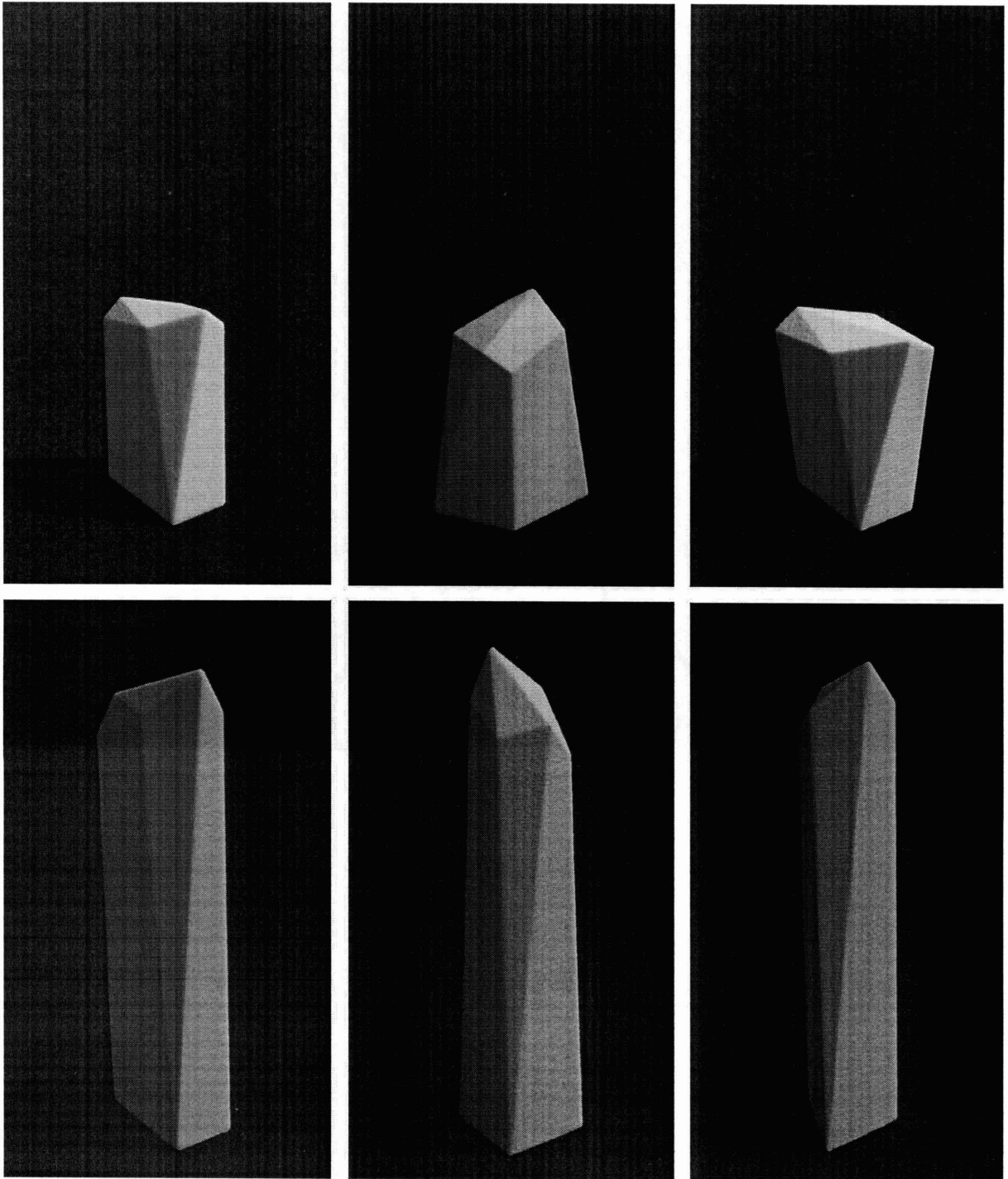


Figure 60: 3D Printed Models Showing Design Variations of the Cira Tower

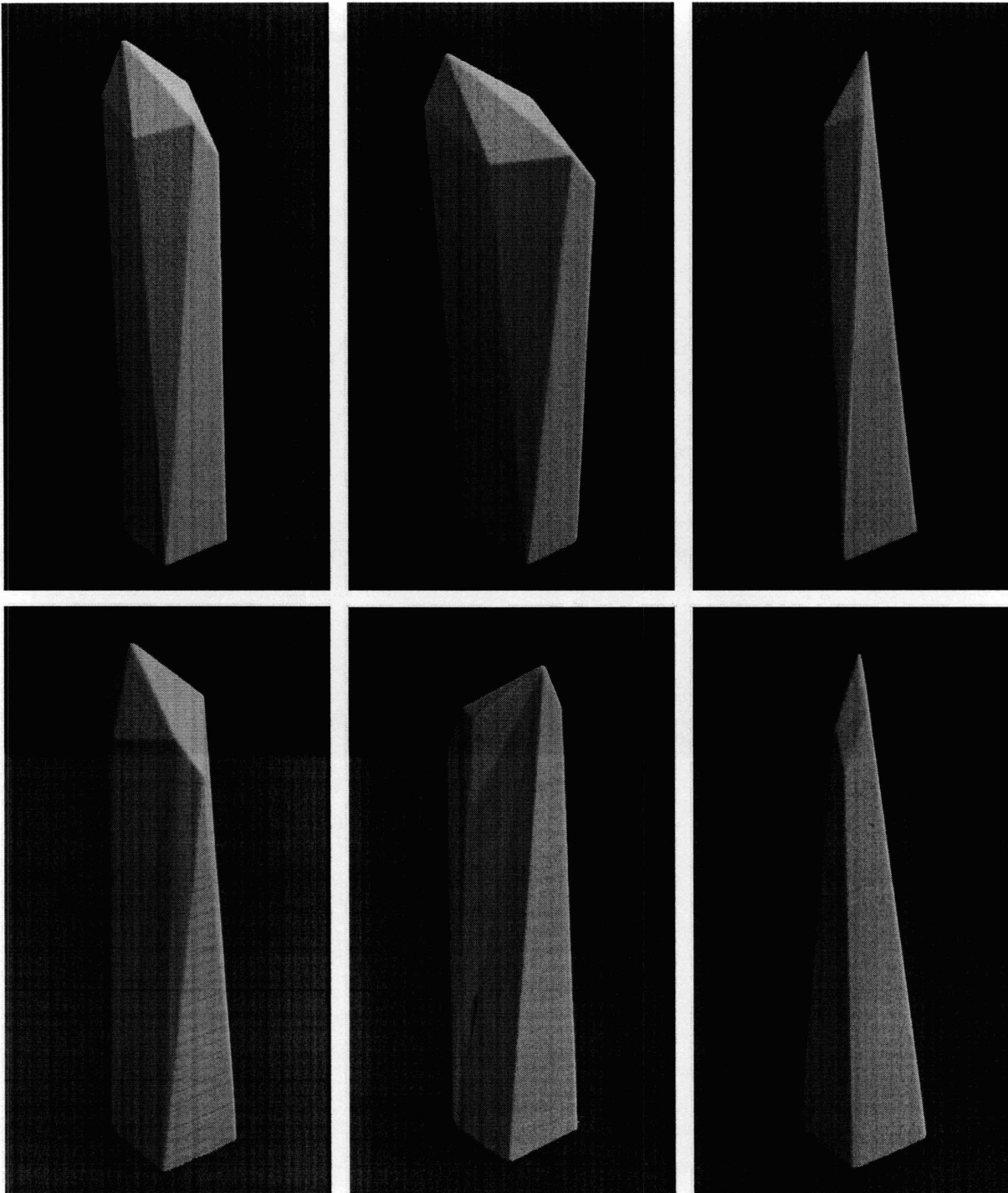
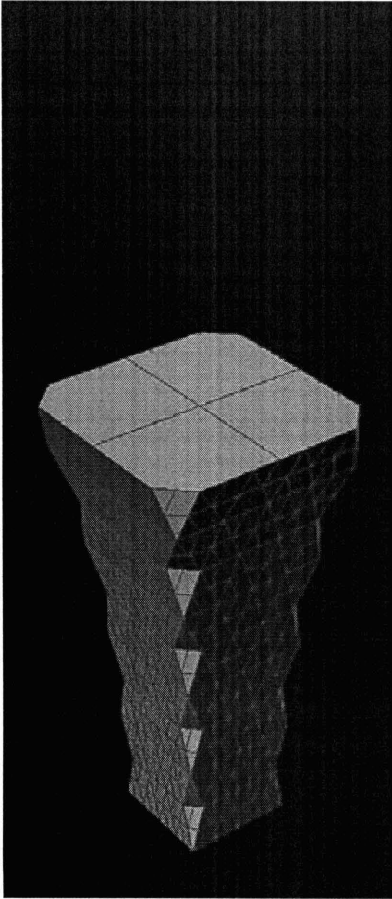
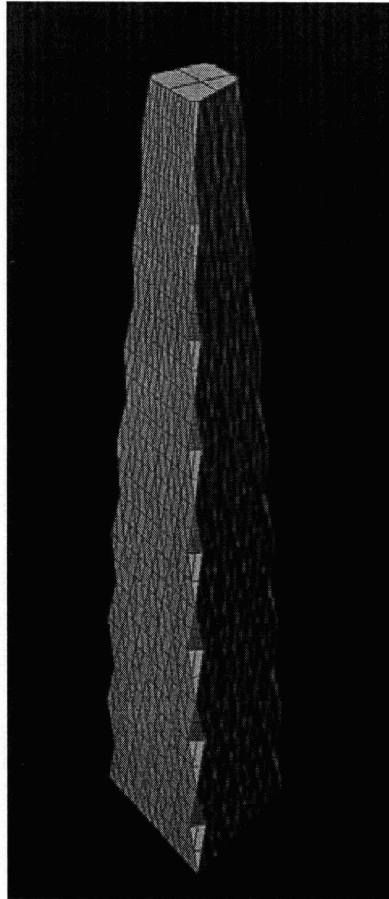


Figure 61: 3D Printed Models Showing Design Variations of the Cira Tower



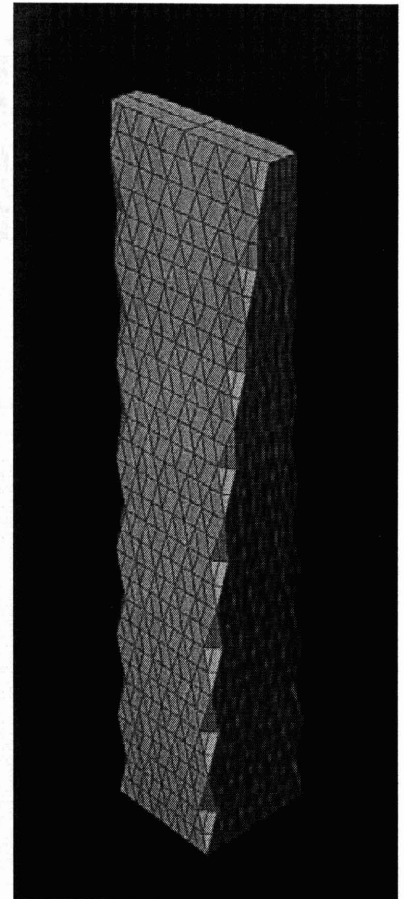
Settings:

Floor Height: 4200
 Floor Number: 45
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1.45
 Top Profile YScale: 1.45
 Rotation Angle: 0



Settings:

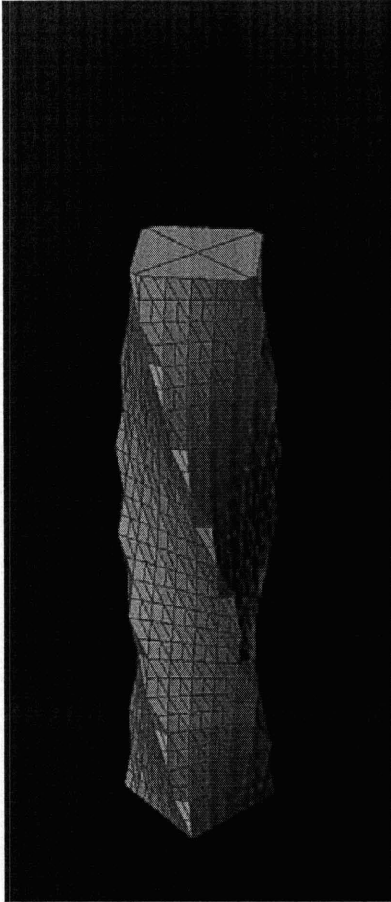
Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.5
 Top Profile YScale: 0.5
 Rotation Angle: 0



Settings:

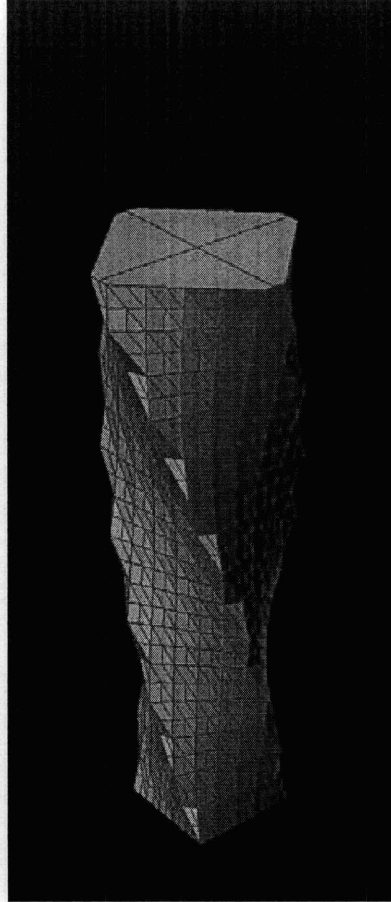
Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.25
 Top Profile YScale: 1.25
 Rotation Angle: 0

Figure 62: Design Variations of the Hearst Corporation Generated by the Proposed System
 These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanation of these rules can also be found in Chapter 4.



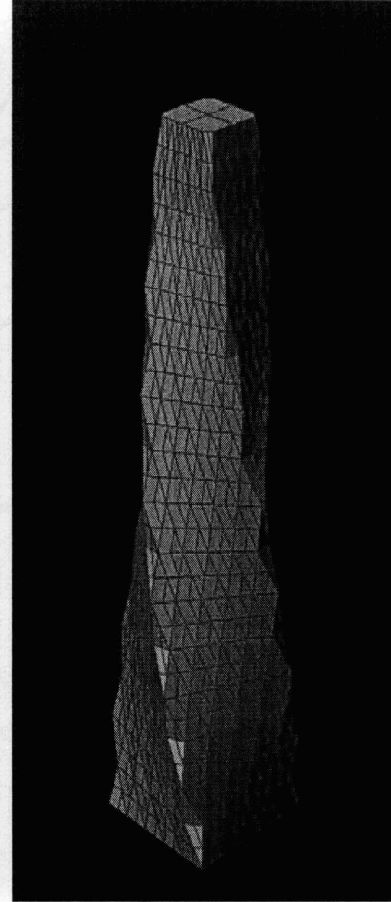
Settings:

Floor Height: 4200
 Floor Number: 60
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1
 Top Profile YScale: 1
 Rotation Angle: 135



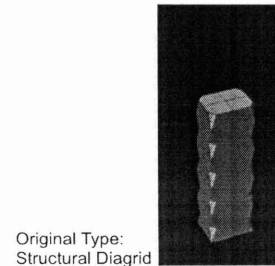
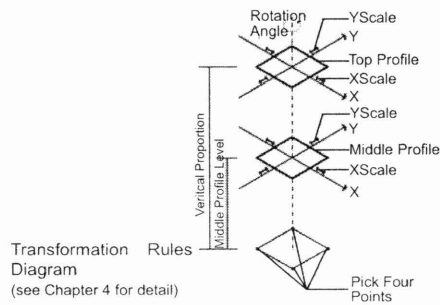
Settings:

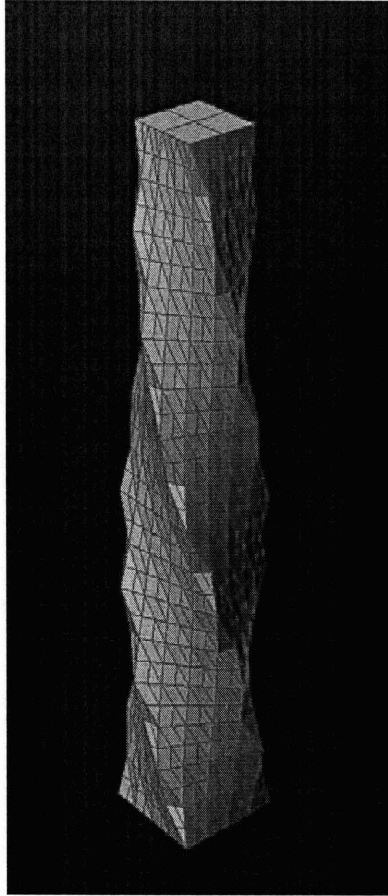
Floor Height: 4200
 Floor Number: 60
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 1.25
 Top Profile YScale: 1.25
 Rotation Angle: 135



Settings:

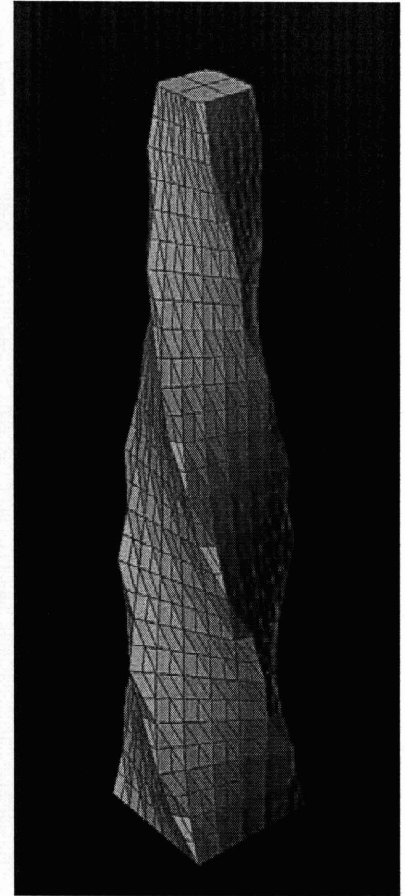
Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.5
 Top Profile YScale: 0.5
 Rotation Angle: 90





Settings:

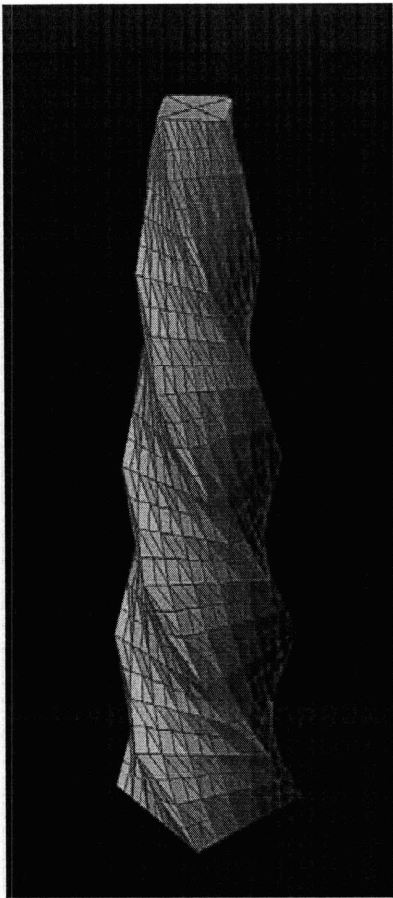
Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.9
 Top Profile YScale: 0.9
 Rotation Angle: 180



Settings:

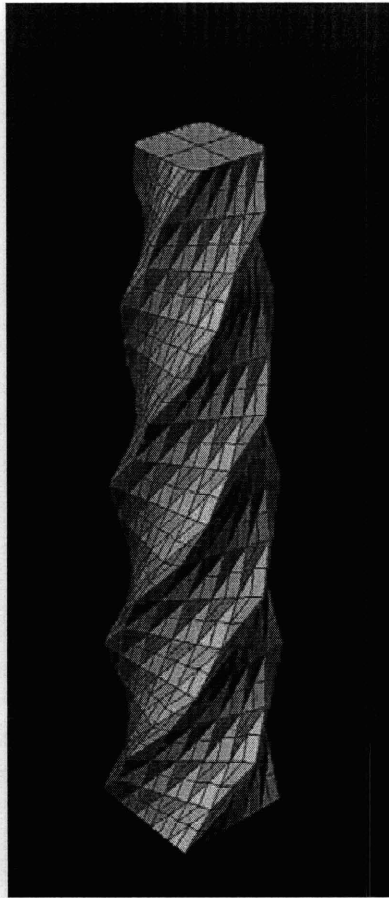
Floor Height: 4200
 Floor Number: 80
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.5
 Top Profile YScale: 0.5
 Rotation Angle: 270

Figure 64: Design Variations of the Hearst Corporation Generated by the Proposed System
 These images show models generated by the proposed system. Settings used in the generation process are identified in each image. Explanations about these settings can be found in Transformation Rules Diagram on the following page. Detail explanation of these rules can also be found in Chapter 4.



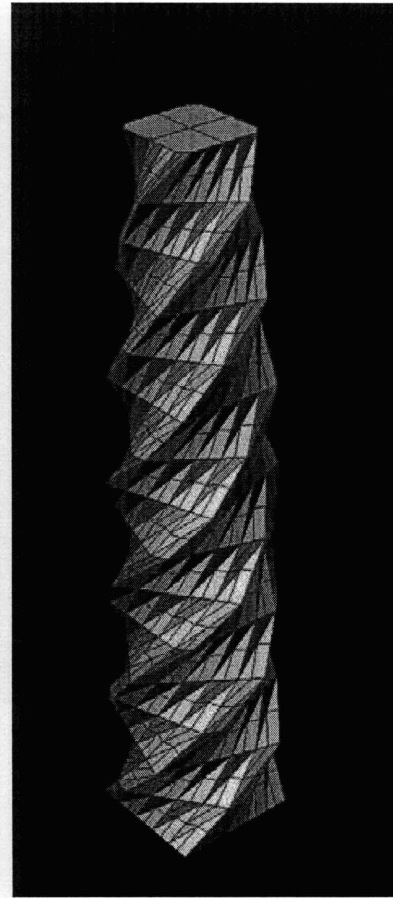
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.45
 Top Profile YScale: 0.45
 Rotation Angle: 360



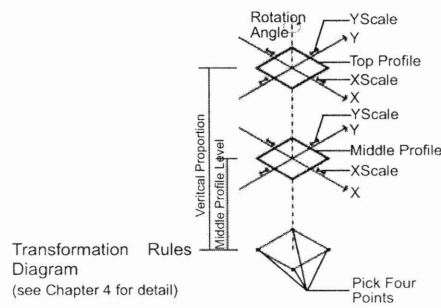
Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.85
 Top Profile YScale: 0.85
 Rotation Angle: 540



Settings:

Floor Height: 4200
 Floor Number: 75
 Middle Profile Level: NA
 Middle Profile XScale: NA
 Middle Profile YScale: NA
 Top Profile XScale: 0.85
 Top Profile YScale: 0.85
 Rotation Angle: 720



Original Type:
 Structural Diagrid

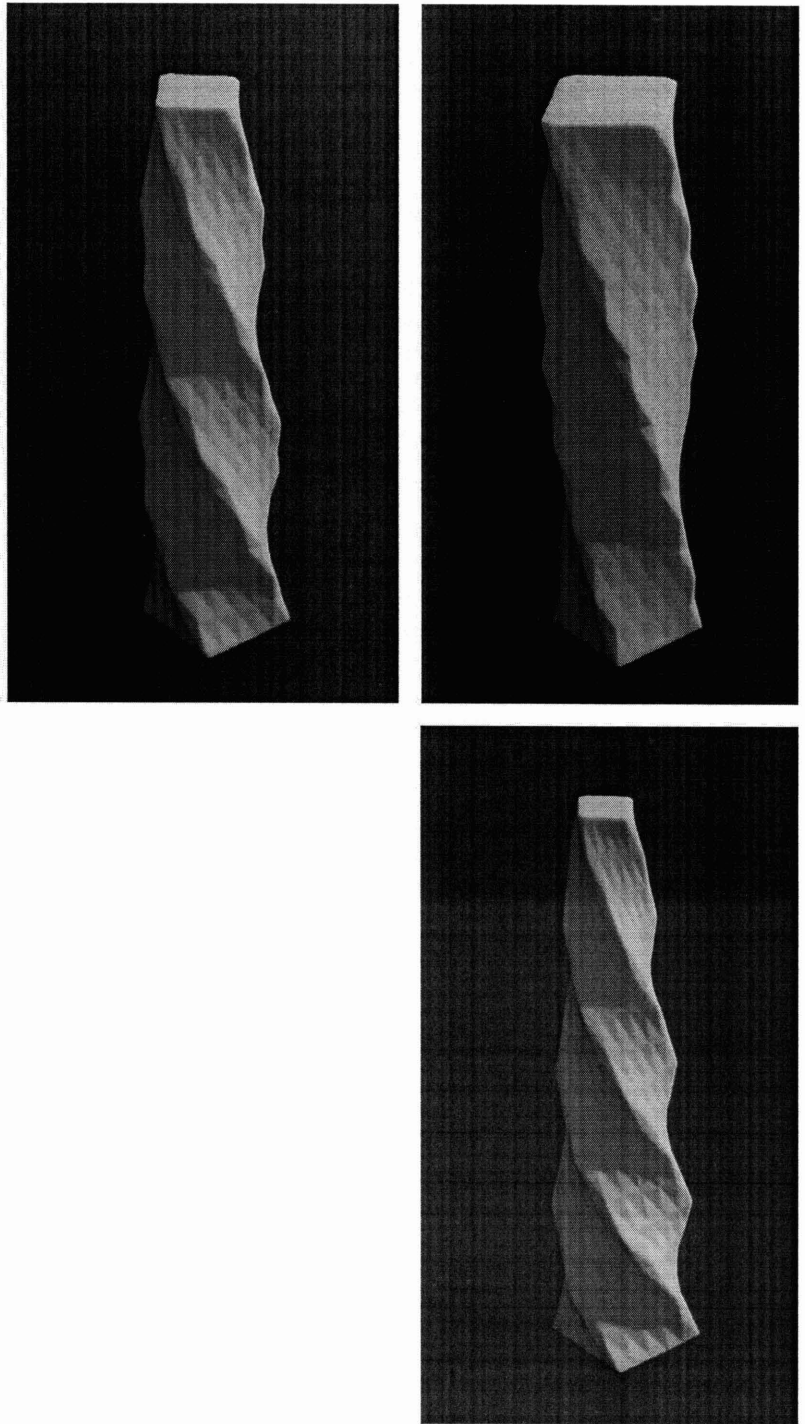


Figure 66: 3D Printed Models Showing Design Variations of the Hearst Corporation

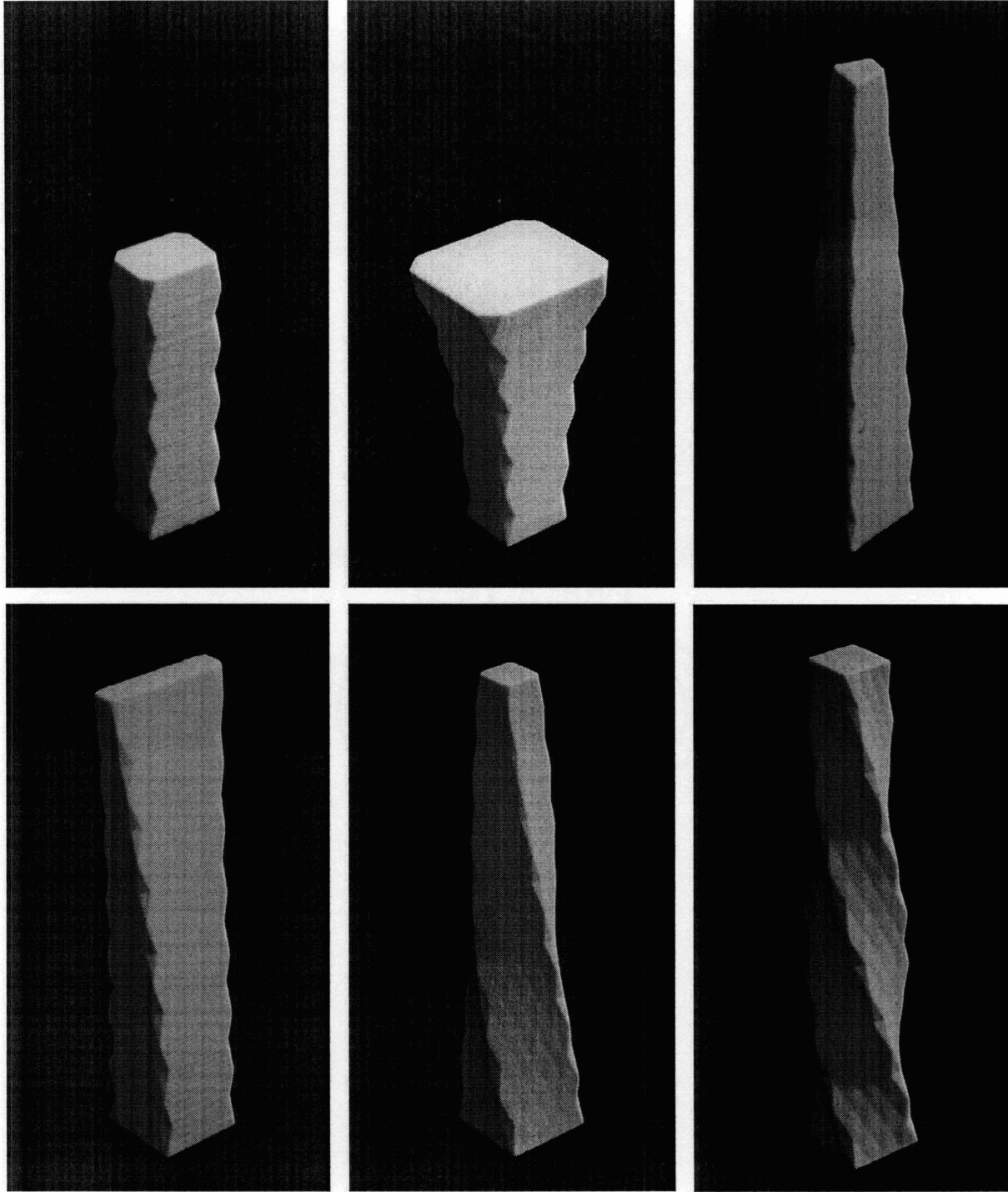


Figure 67: 3D Printed Models Showing Design Variations of the Hearst Corporation

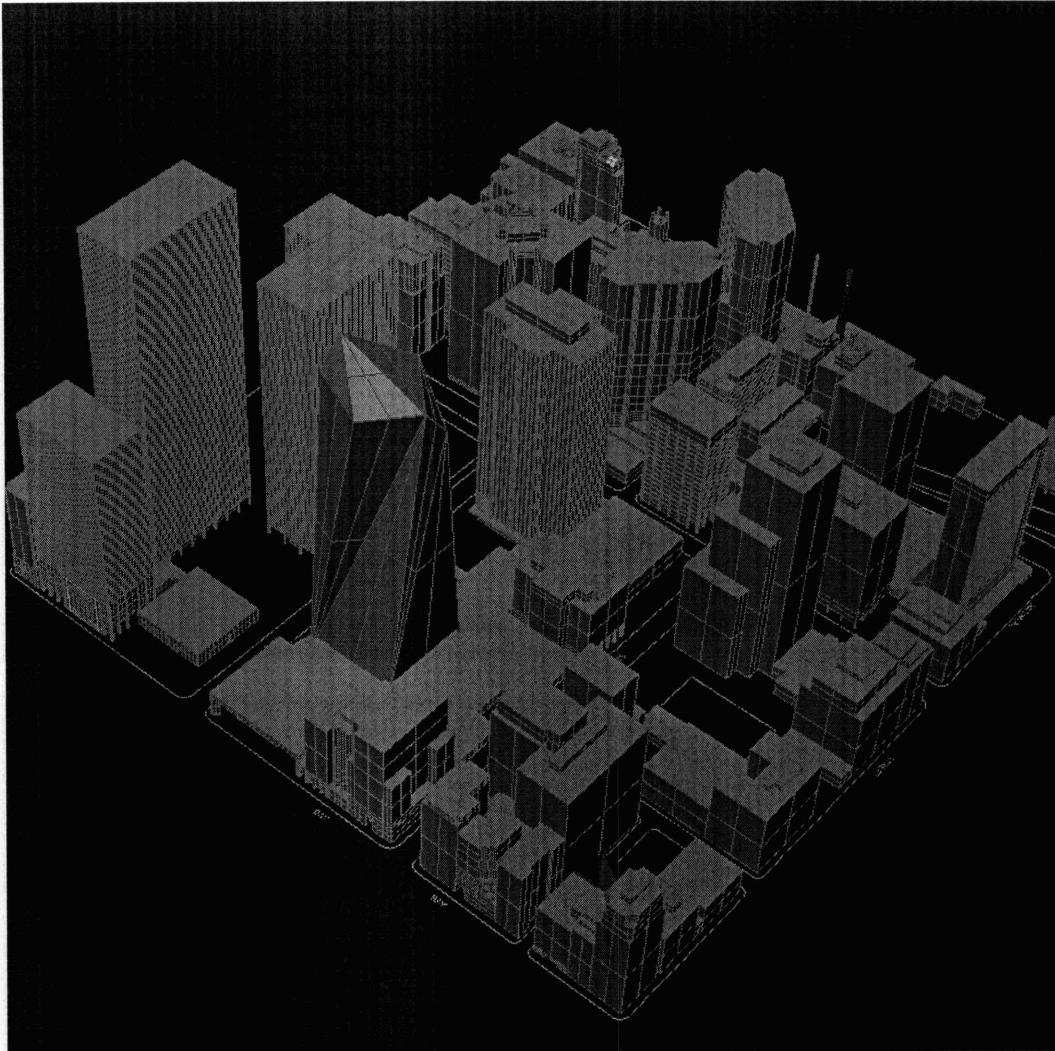


Figure 68: Constructing Massing in a Site Context
The diagram shows the way to construct massing in a selected site. Users can interactively view the construction progress in different angles.

5.1.2 Constructing Massing in a Site Context

Figure 68 demonstrates one of the applications of the proposed system: to interactively build the massing of towers in a site context. Users have opportunities to view the progressive construction of the tower in a three-dimensional environment. On the following pages, some variations on a same site are presented (Figure 69).

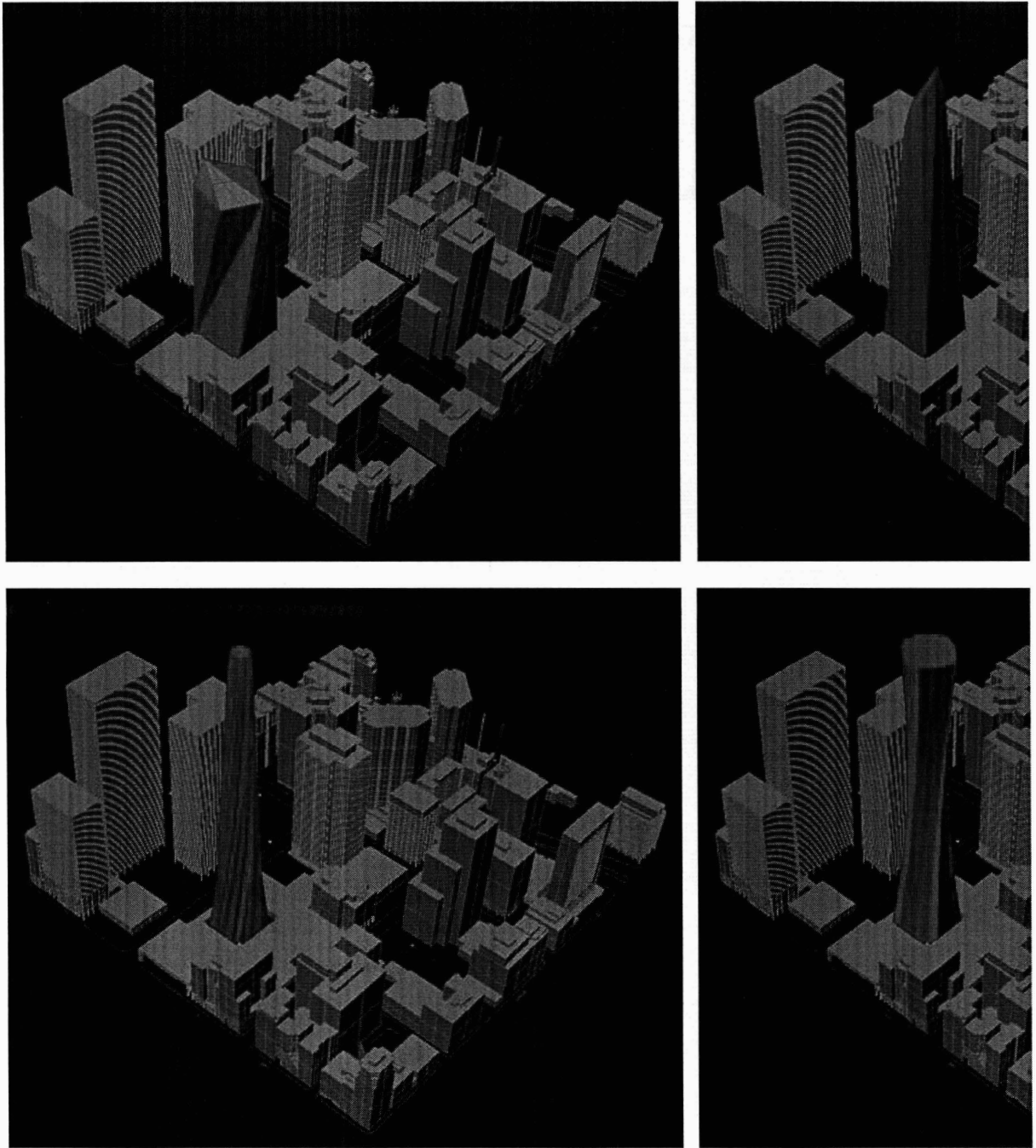


Figure 69: Constructing Variations on a Same Site
One of the advantages of the program is to allow users to construct many variations on the same site and they can compare these variations while having these variations at hand.

Figure 10-10: 3D Model of the City of Chicago

1

Figure 10-11: 3D Model of the City of Chicago

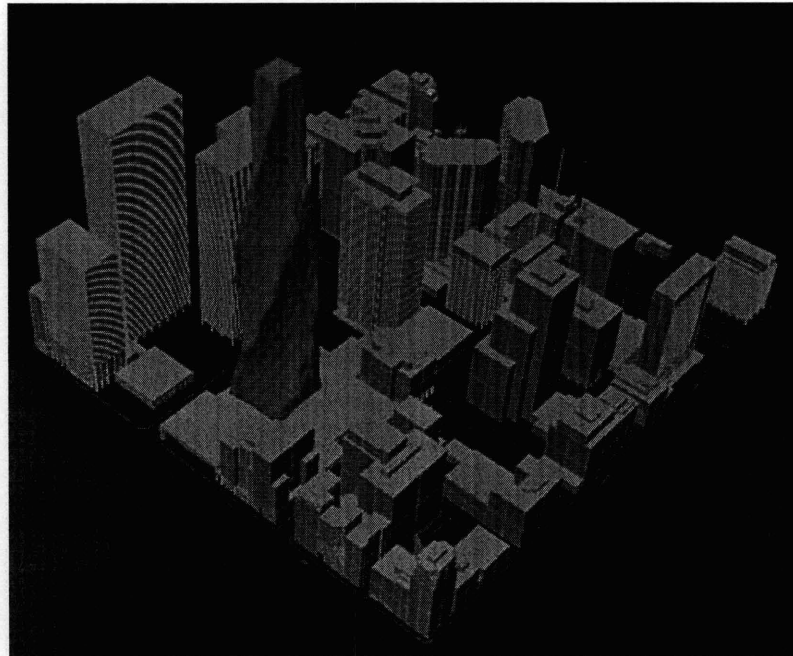
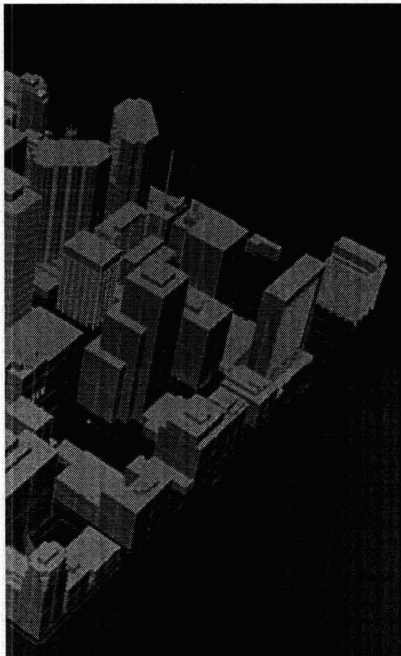
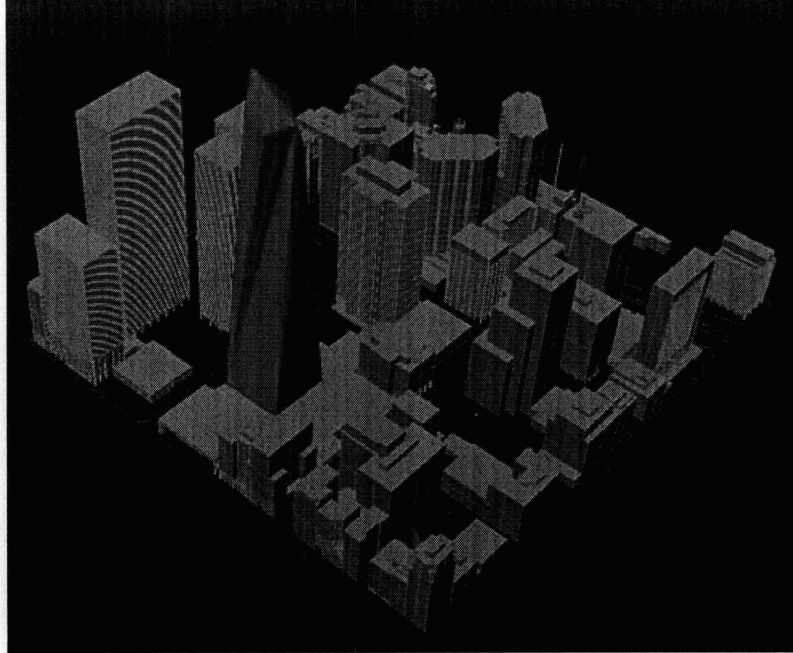
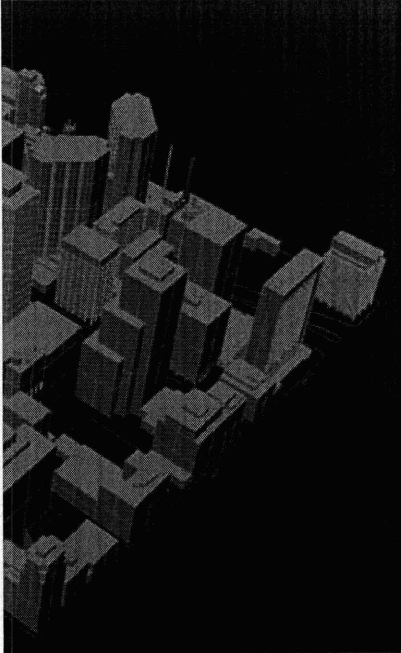
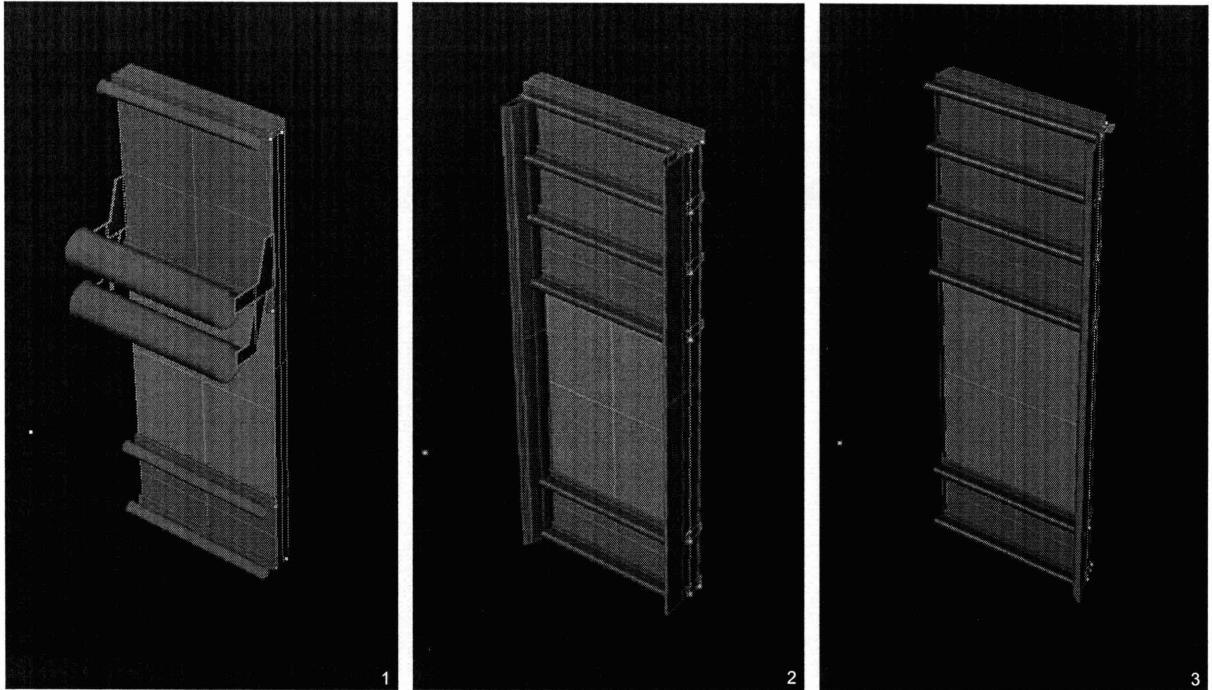




Figure 71: 3D Printed Models Showing the Forrest of Towers of the Variations Shown at Previous pages



Types of Details:

1. Reinforced Horizontality
2. Reinforced Verticality-Regular
3. Reinforced Verticality-Corner
4. Reinforced Verticality-Mechanical
5. Structural Diagrid

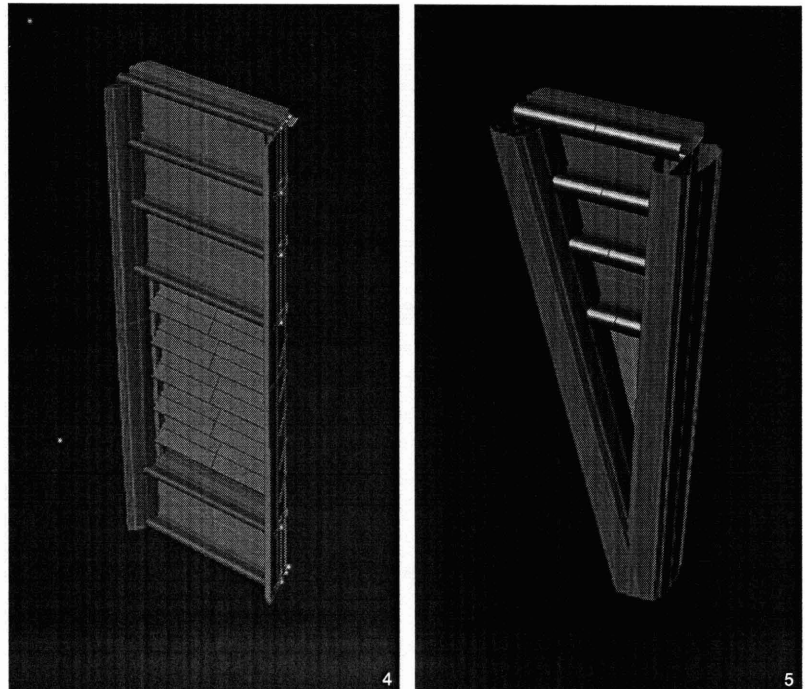


Figure 72: Types of Details
 These images show five types of details that were embedded into the proposed system. These details can be used independently or combined together for a design project. See following section for the examples of their applications.

5.2 Detail

5.2.1 Types of Details

Figure 72 shows five types of details that were embedded in the systems. They are: the Reinforced Horizontality, the Reinforced Verticality-Regular, the Reinforced Verticality-Corner, the Reinforced Verticality-Mechanical, and the Structural Diagrid. The Reinforced Horizontality detail features expressive horizontal mechanical louvers and seamless glass panels on vertical positions. The Reinforced Verticality-Regular detail is characterized by reinforced vertical mullions. The Reinforced Verticality-Corner detail has a reinforced vertical mullion on one side and a seamless connection on the other. The Reinforced Vertical-Mechanical provides mechanical louvers and is used in a location behind which mechanical rooms are located. The Structural Diagrid detail is a triangulated structural component that is used on structural skins. All types of details are constructed by using the Associative Relationship between Position Points and Adjacent Points method that was explained in Chapter 4.

5.2.2 Constructing Details on a Selected Surface

Figure 73 on the following page shows one important application of the system: constructing details in a selected surface. It allows user to pick one single surface and then identify the lowest and highest points; then the program will find level lines and references points on these lines to construct details components. The main idea behind this

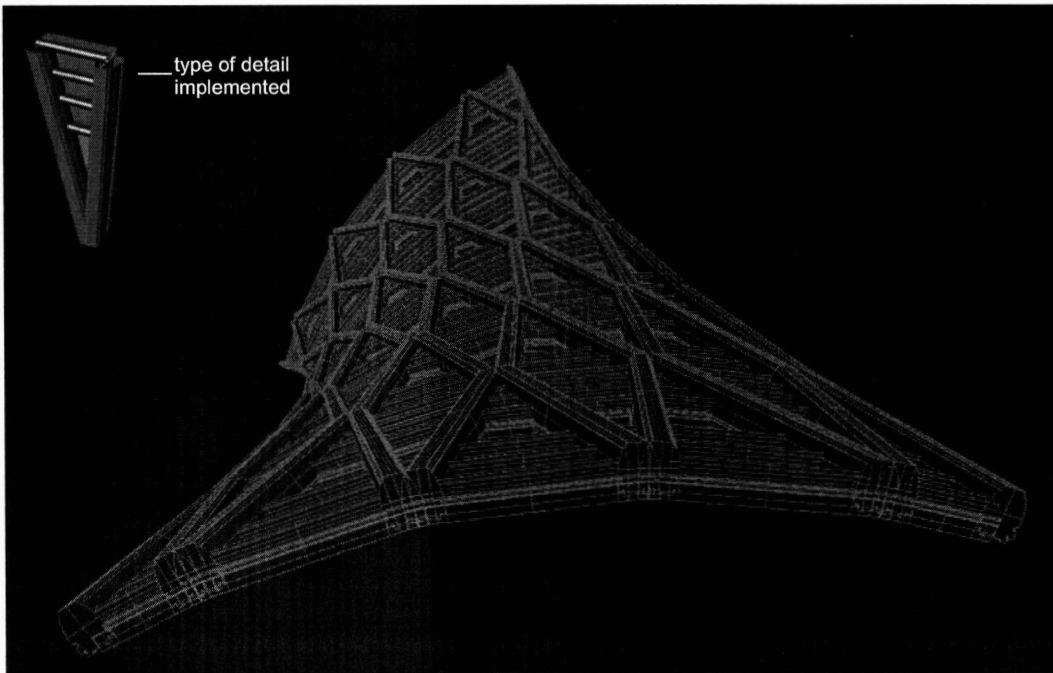
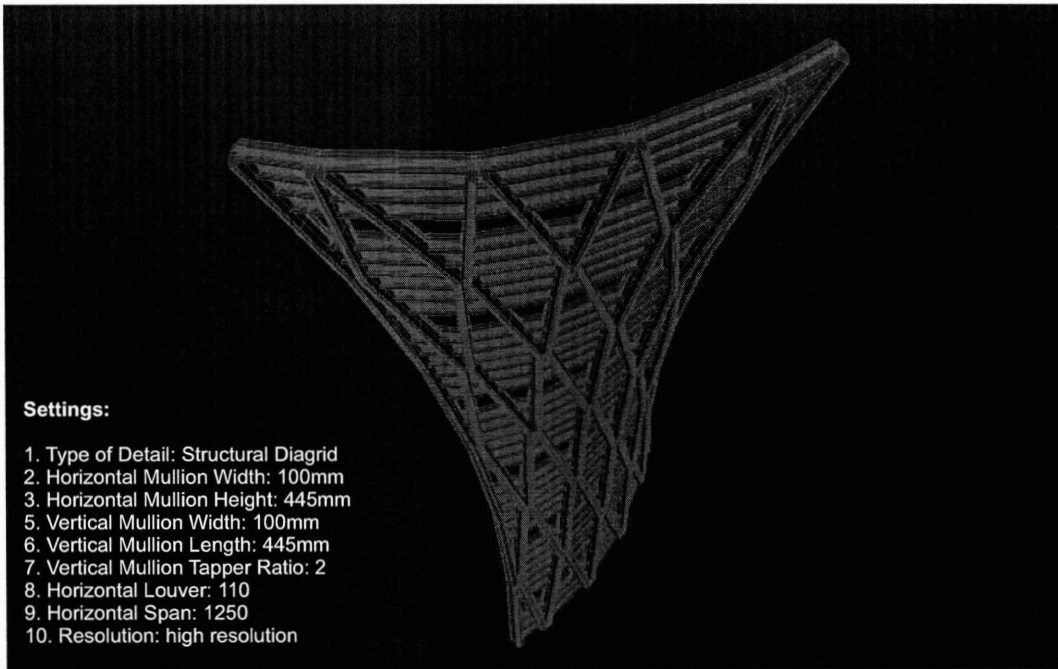
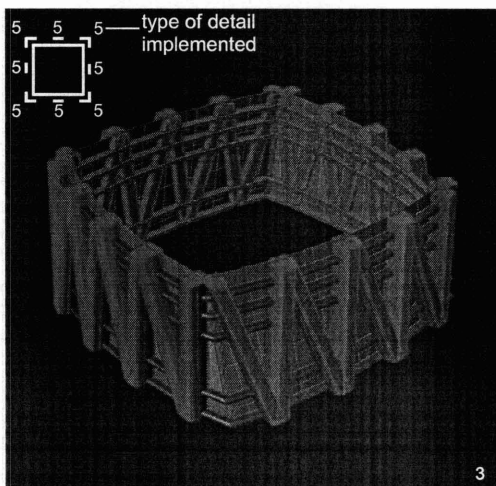
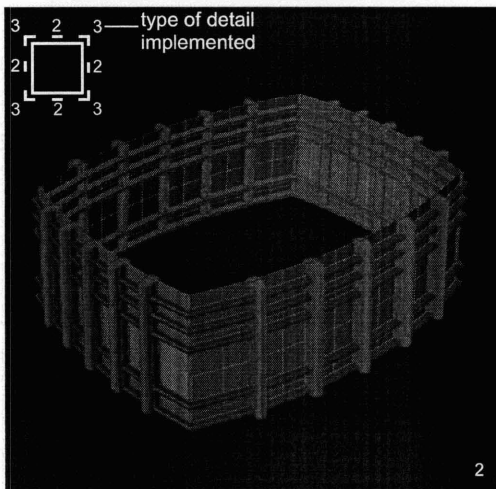
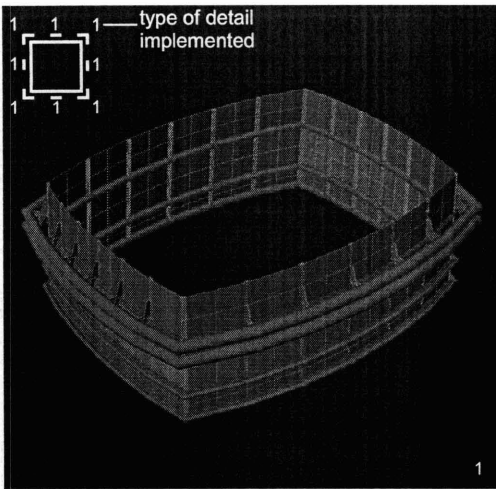


Figure 73: Diagrid Constructed on a Surface
The program allows users construct the whole surface of a façade to evaluate design intention within a reasonable time frame. Settings used are shown on each image.



application is to allow user construct the whole surface of a façade to evaluate design intention within a reasonable time frame.

5.2.3 Constructing Details on Selected Levels

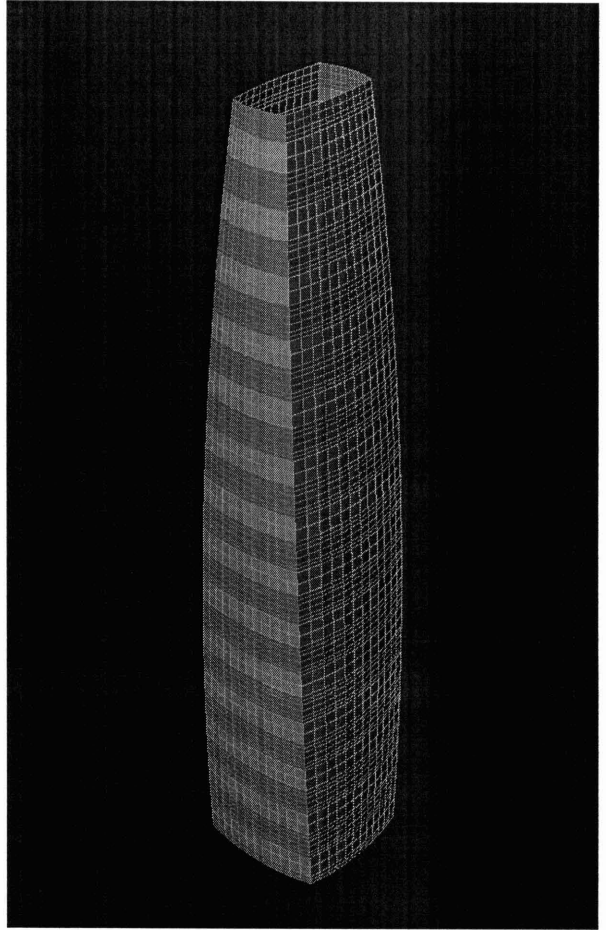
Efficiency is one of the major concerns for writing a program. It is important that the program can accomplish tasks within a reasonable time frame; otherwise, it will lose its practical values. In order to efficiently construct a highly detailed tower, the proposed system provides an option for users to construct details on selected levels. By doing so, users can construct the whole tower (construct one level after the other) within two hours comparing to ten without using this option. Another benefit of this option is to allow users to construct different details on different levels so that they can visually compare different types of details on the same towers. Figure 74 shows three examples of constructing three different types of details by using this option.

Types of Details:
(images of detail types can be seen on Chapter 5.2.1)

1. Reinforced Horizontality
2. Reinforced Verticality-Regular
3. Reinforced Verticality-Corner
4. Reinforced Verticality-Mechanical
5. Structural Diagrid

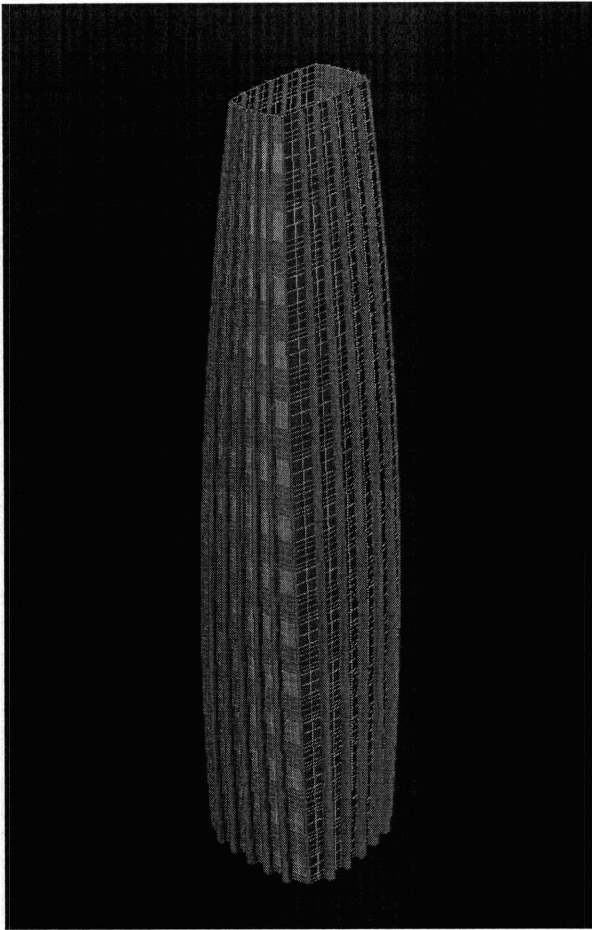
- I. Example One: Horizontality
- II. Example Two: Verticality
- III. Example Three: Diagrid

Figure 74: Constructing Details on Selected Levels
For the efficiency is concerned, the program provides an option for users to construct details on selected levels. They can also construct different details on different levels to compare these details on the same tower. Types of details used are indicated on each image.

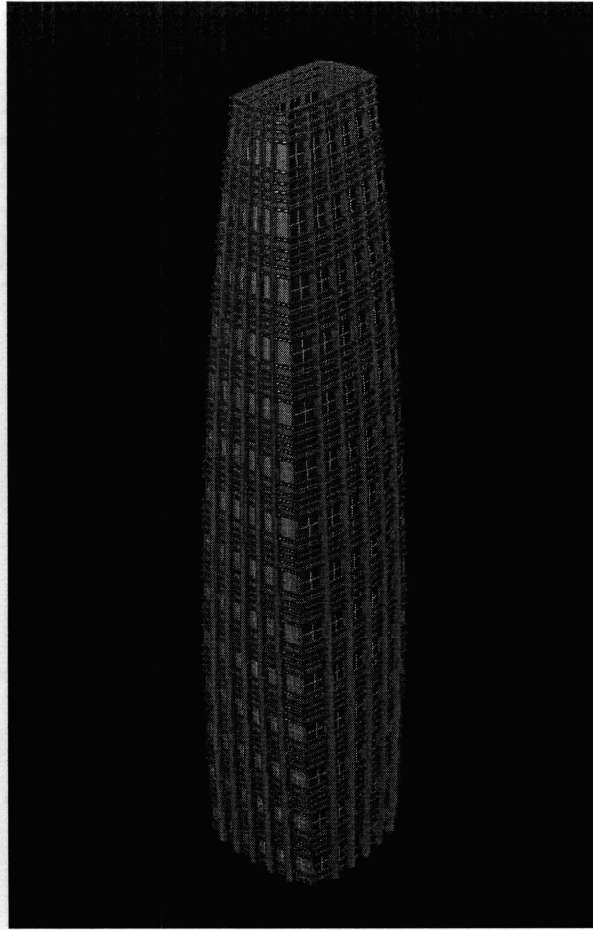


Low Resolution Setting

Figure 75: Construct Detail in Different Resolution Settings
These images show the examples of a tower generated by using three different resolution settings.



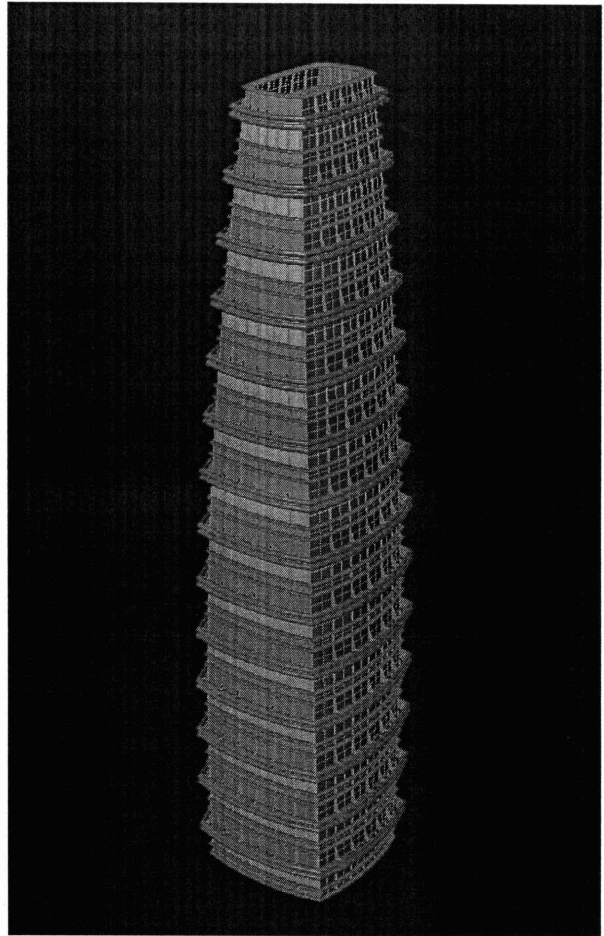
Medium Resolution Setting



High Resolution Setting

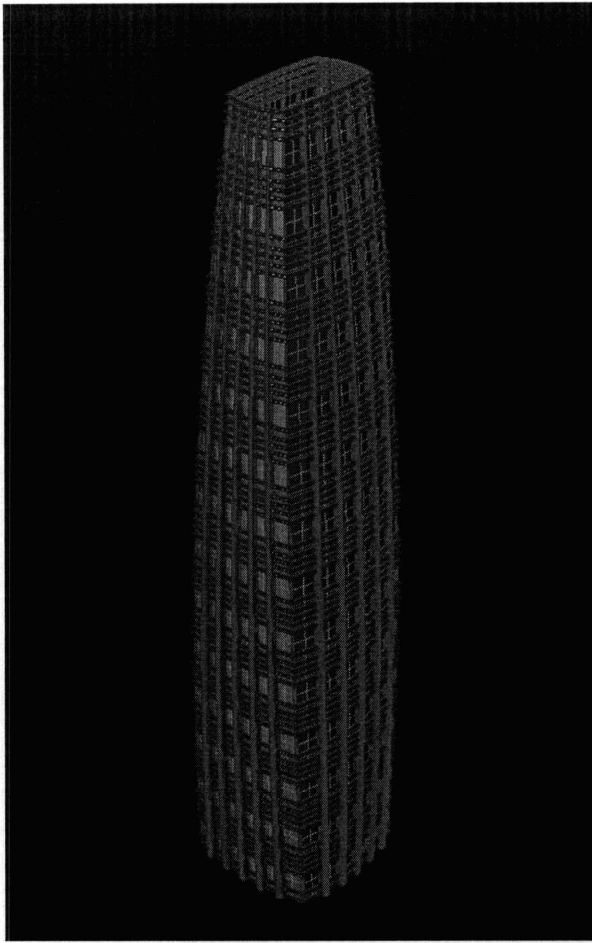
5.2.4 Construct Details in Different Resolution Settings

Figure 75 shows three different resolution settings in the system. The low resolution setting only constructs glass and spandrel panels. The medium resolution offers more articulated details of the tower and its elevation composition starts to be read. The high resolution setting constructs all details that reflect the actual conditions.

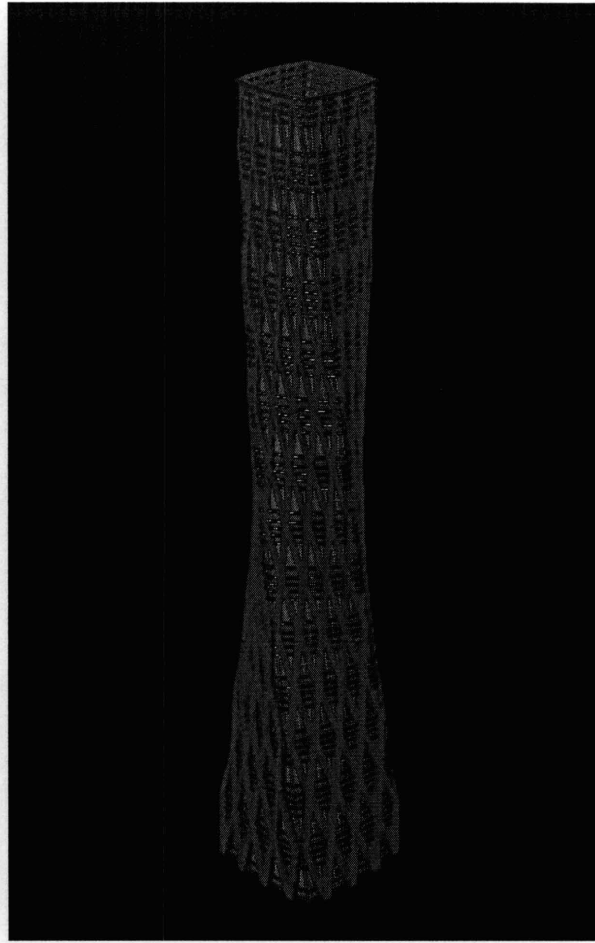


No-Name Tower

Figure 76: Incorporation of Massing and Details
These images show three new designs of skyscrapers that were generated by the proposed system. All details are constructed.



Drum-Shaped Tower



New Twisted Tower

5.3 Incorporation of Massing and Detail

Figure 76 shows three different resolution settings in the system. The low resolution setting only constructs glass and spandrel panels. The medium resolution offers more articulated details of the tower and its elevation composition starts to be read. The high resolution setting constructs all details that reflect the actual conditions.

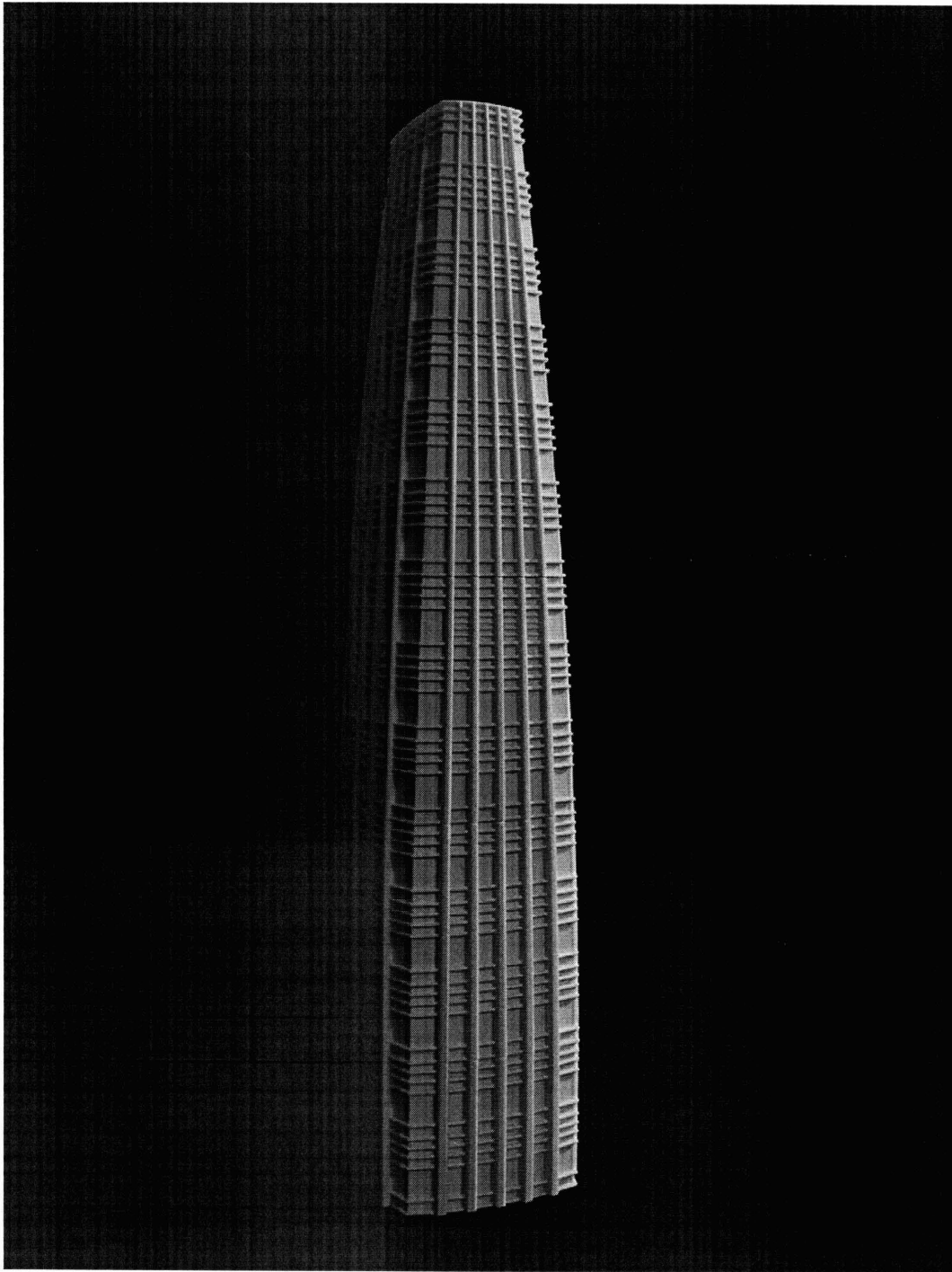


Figure 77: 3D Printed Models Showing a New Design: the Drum-Shaped Tower

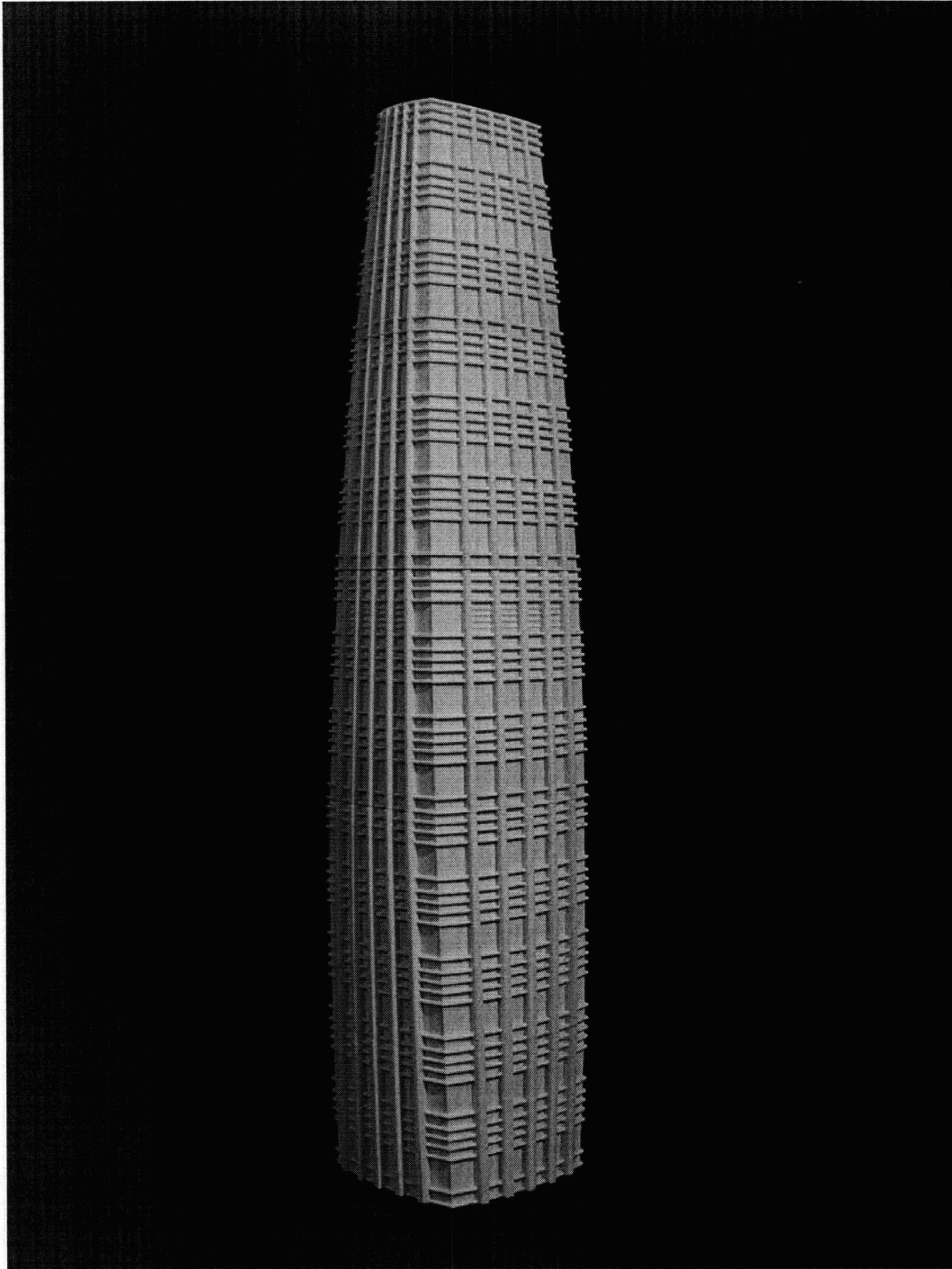


Figure 78: 3D Printed Models Showing a New Design: the Drum-Shaped Tower

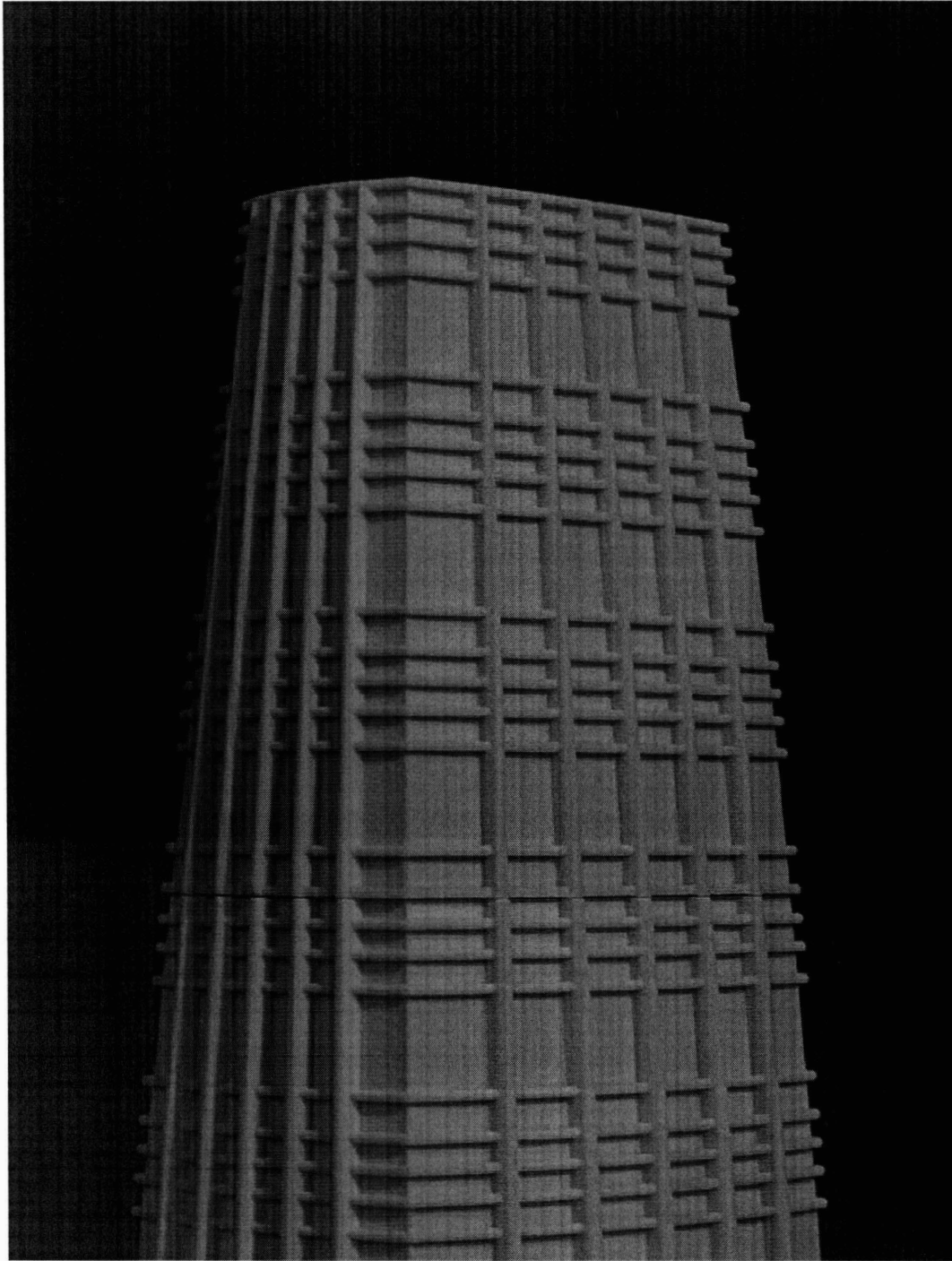


Figure 79: 3D Printed Models Showing the Details on the Top Levels of the Drum-Shaped Tower

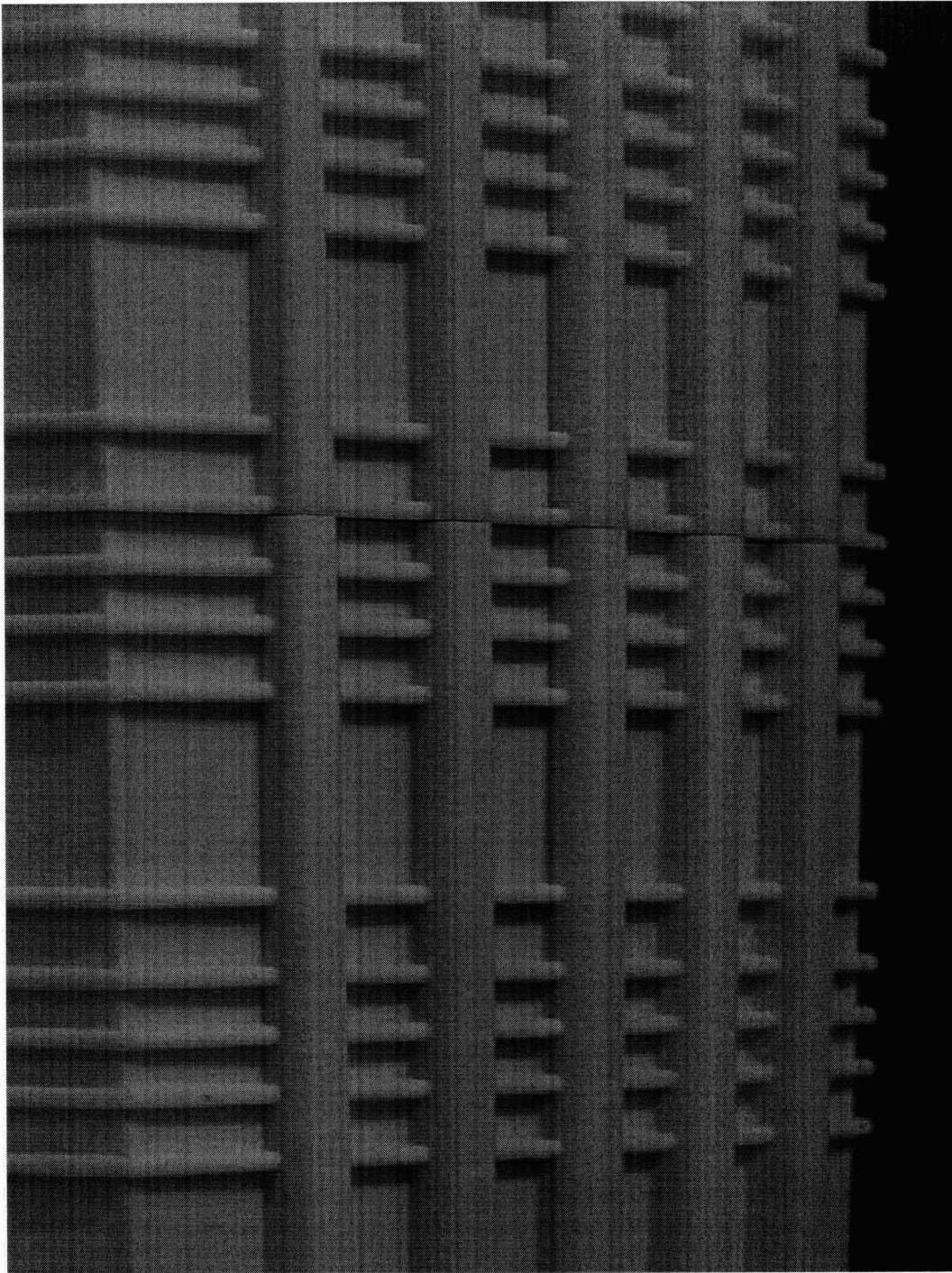


Figure 80: 3D Printed Models Showing the Details on the lower Levels of the Drum-Shaped Tower

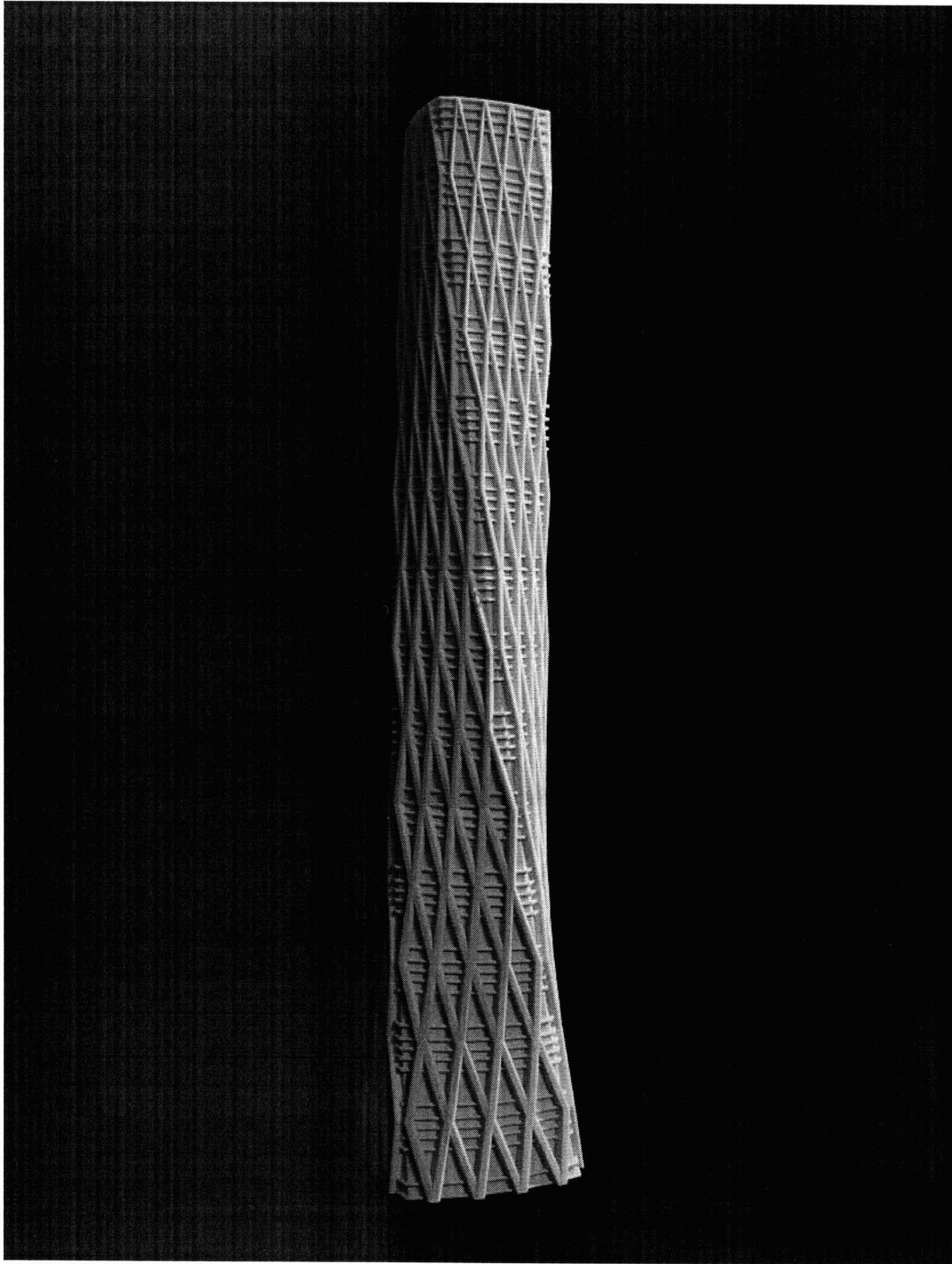


Figure 81: 3D Printed Models Showing a New Design: the New Twisted Skyscraper

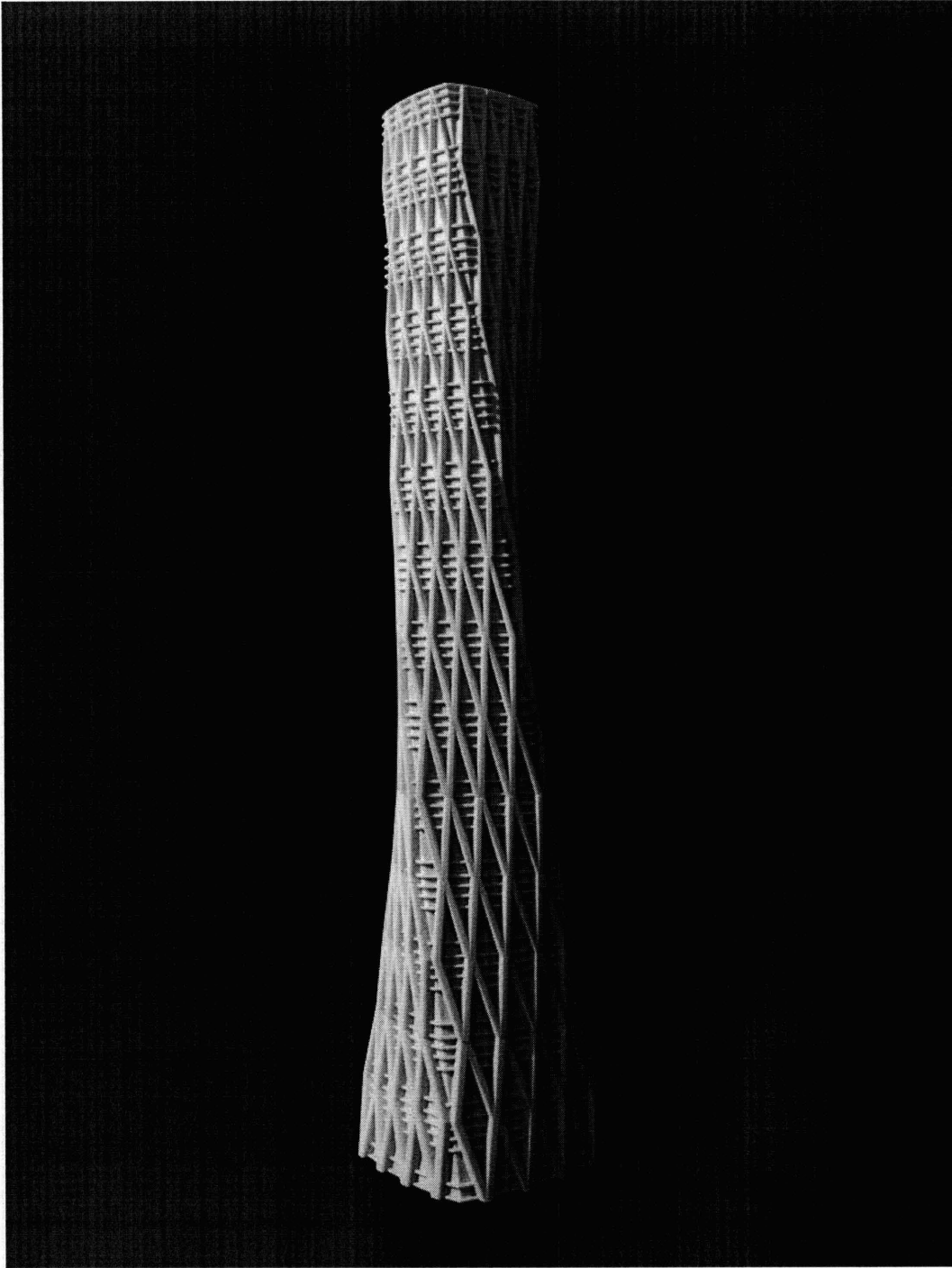


Figure 82: 3D Printed Models Showing a New Design: the New Twisted Skyscraper

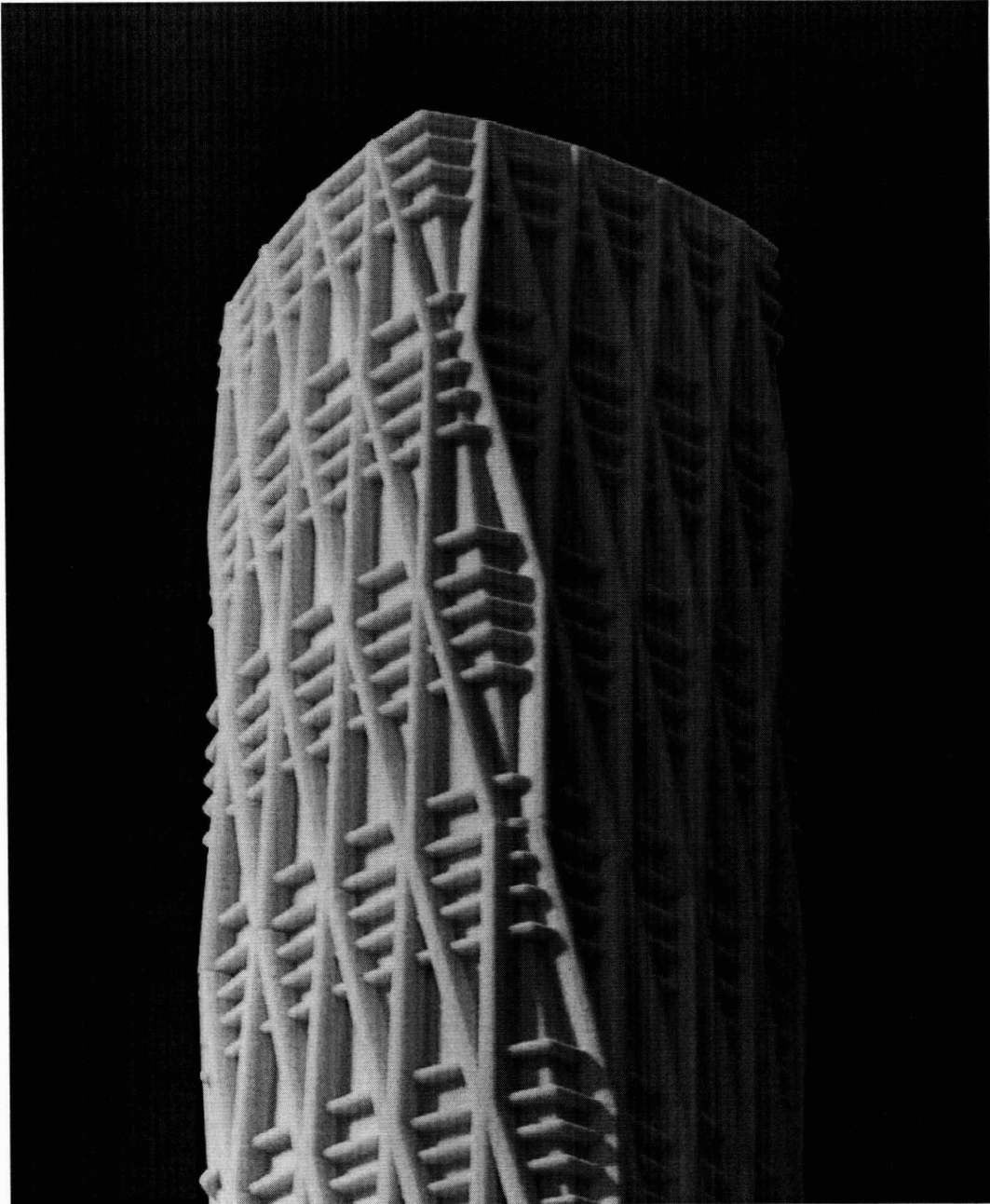


Figure 83: 3D Printed Models Showing the Details on the Top Levels of the New Twisted Skyscraper

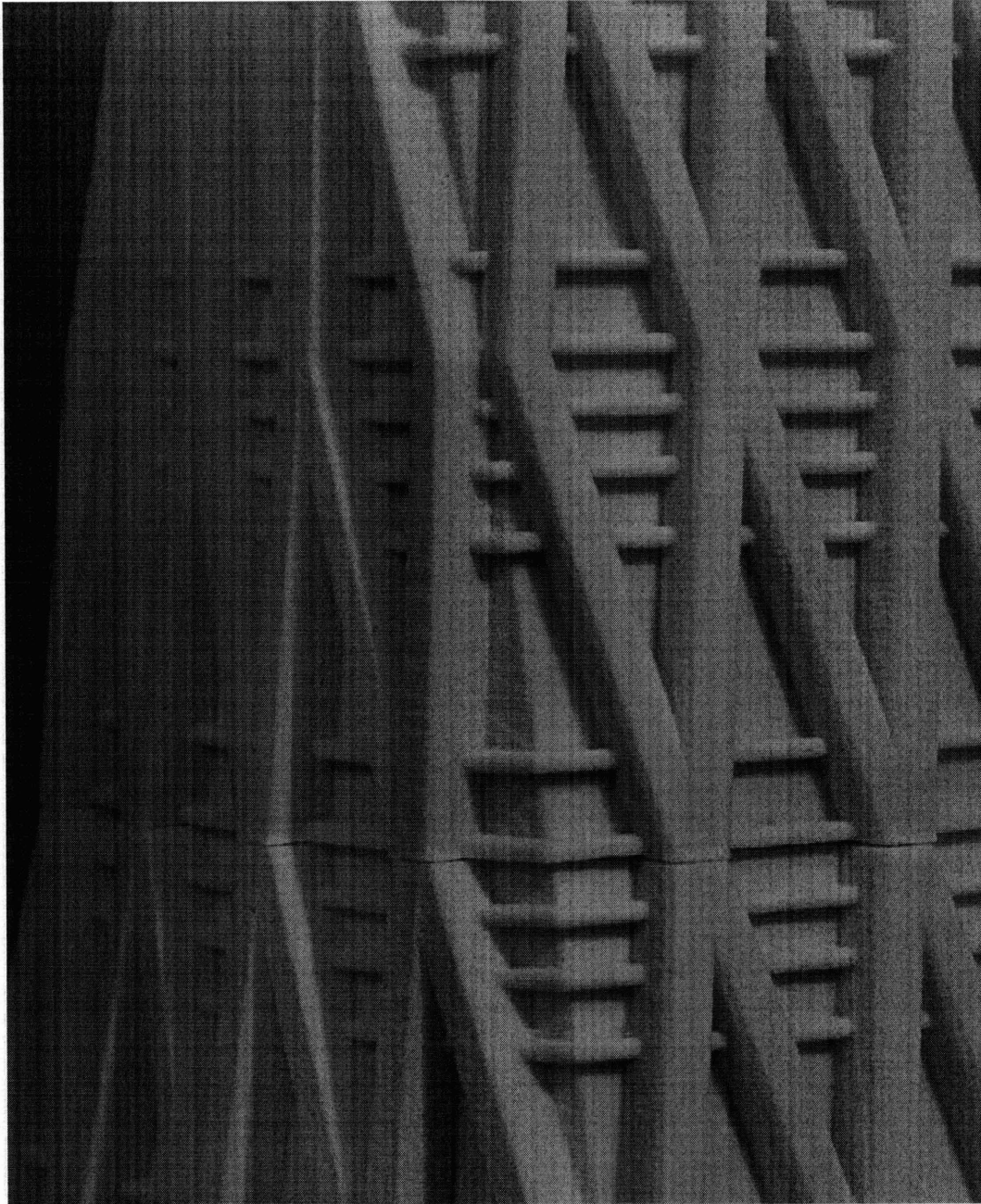


Figure 84: 3D Printed Models Showing the Details on the Lower Levels of the New Twisted Skyscraper

Chapter 6: Conclusion and Future Work

6.1 Conclusion--Towards a New Digital System for Skyscraper Design

The objective of this work has been to develop a digital system to help architects design skyscrapers in the early stage of the design process. This system implements embedding methods and is capable of handling both the massing and details in the computer generated design of skyscrapers. Computational methods were developed and became the mechanism to ensure that the computer could capture design intentions as well as physical conditions that designers usually encounter in the design process. Experiments were taken to test the validity and the range of application of the system in re-constructing the original design as well as in generating variations. The result of these experiments reflects that, by automating the form finding and component generating processes, architects can work more efficiently and effectively. The result also demonstrates that embedding methods are appropriate for carrying previous experiences and expertise into the new design of skyscrapers.

On a larger scale, the thesis shows the importance of automating certain portions of the design process and the benefits of doing so: architects can avoid repetitive work and concentrate on more creative issues. Also, the thesis demonstrates that architects can construct better digital systems to handle their particular design tasks instead of relying on most of the systems that were developed by engineer and computer scientists.

6.2 Future Work – To Expand the Capacity of the System

The proposed system was developed through in-depth studies of some selected projects and by doing experiments to re-construct these projects and their variations. To some extent, the developed methods can be generalized to handle many different design conditions that were not identified in the thesis. This system can be tested by implementing it in practical projects. In such a way, the potentials of this system can be explored and its limitations can be recognized so that adjustments can be made accordingly.

The initial scope of the system focused on the exterior skins of skyscrapers. It did not take into consideration the conditions behind these exterior skins, such as connections between the skins and the structure or the functional programs in the interior. Also, this system does not provide a mechanism to test the structural and ecological performances of the digital models it generates. These new features would be very useful; I hope that new versions of the system will be developed to include these analytical functions.

Appendixes:

Appendix A: VB.Net

This appendix contains codes in VB.Net program that was used to construct the user interface of the program. (Only the fundamental parts of the codes are shown.)

VB_UseRhinoScriptCommand.VB

```
Imports RMA.Rhino
Imports RMA.OpenNURBS

Public Class VB_UseRhinoScriptCommand
Inherits RMA.Rhino.MRhinoScriptCommand
    Public Overrides ReadOnly Property EnglishCommandName() As String
        Get
            Return "DigitalSkyscraper"
        End Get
    End Property
    'RunCommand is called when the user enters the "DigitalSkyscraper" command
in Rhino
    Public Overrides Function RunCommand(ByVal context As IRhinoCommandContext)
        As IRhinoCommand.result
        Dim command_result As IRhinoCommand.result
        command_result = IRhinoCommand.result.nothing
        Dim myForm As New Digital_Skyscraper
        myForm.Show()
        Return command_result
    End Function
End Class
```

Digital_Skyscraper.VB

```
Private Sub BottonGenerate_Tower_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles BottonGenerate.Click
    Dim strComponentSyle As String
    Dim strNumberOfHorizontalMullions As String
    Dim strHorizontalMullionsStyle As String
    Dim strVerticalMullionWidth As String
    Dim strVerticalMullionLength As String
    Dim strHorizontalMullionWidth As String
    Dim strHorizontalMullionHeight As String
    Dim strHorizontalLouvreSize_Width As String
    Dim strVerticalMullionWidthTapperRatio As String
```

```

Dim strArchitectureStyleForcombination As String
Dim strStartLevel As String
Dim strEndLevel As String
If CheckBox_Component_HKIFC_1.Checked Then
    strComponentSyle = "Cesar_HKIFC"
End If
    If CheckBox_Component_Structure_Setting.Checked And CheckBox_Comp-
onent_HKIFC_1.Checked Then
        strComponentSyle = "Cesar_HKIFC"
        strArchitectureStyleForcombination = "Mini_Skyscraper"
    End If
    If CheckBox_Component_Diagrid_1.Checked Then
        strComponentSyle = "Mini_Skyscraper"
    End If
    If CheckBox_Component_Structure_Setting.Checked And CheckBox_Comp-
onent_Petronas_1.Checked Then
        strComponentSyle = "Cesar_Petronas"
        strArchitectureStyleForcombination = "Mini_Skyscraper"
    End If
    If CheckBox_Component_Structure_Setting.Checked = False Then
        strArchitectureStyleForcombination = "Null"
    End If
    If CheckBox_Component_Petronas_1.Checked Then
        strComponentSyle = "Cesar_Petronas"
    End If
    strNumberOfHorizontalMullions = TextBoxNumberOfMullions.Text
    strHorizontalMullionsStyle = ComboBoxMullionArrangement.SelectedItem
    strVerticalMullionWidth = TextBoxVerticalMullionWidth.Text
    strVerticalMullionLength = TextBoxVerticalMullionLength.Text
    strHorizontalMullionWidth = TextBoxHorizontalMullionWidth.Text
    strHorizontalMullionHeight = TextBoxHorizontalMullionHeight.Text
    strHorizontalLouvreSize_Width = TextBoxHorizontalLouvreSize.Text
    strVerticalMullionWidthTapperRatio = TextBox_VerticalMullionTapperRa-
tio.Text
    strStartLevel = TextBoxSelectedLevelFrom.Text
    strEndLevel = TextBoxSelectedLevelTo.Text
    Dim strComponentHorizontalSpan
    strComponentHorizontalSpan = TextBoxComponentHorizontalSpan.Text
    RhUtil.RhinoApp.RunMenuScript("-LoadScript C:\thesis\Run\DigitalSky-
scraper_Component_MiniSkyscraper.rvb " + strComponentSyle + " " + strNumberOf-
HorizontalMullions + " " + strHorizontalMullionsStyle + " " + strVerticalMul-
lionWidth + " " + strVerticalMullionLength + " " + strHorizontalMullionWidth +
" " + strHorizontalMullionHeight + " " + strHorizontalLouvreSize_Width + " " +
strComponentHorizontalSpan + " " + strVerticalMullionWidthTapperRatio + " " +
strArchitectureStyleForcombination + " " + strStartLevel + " " + strEndLevel +
" ")
End Sub

Private Sub ButtonGenerateMassing_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ButtonGenerateMassing.Click

```

```

Dim strArchitecturalSyle As String
Dim strMiddleProfileLevel As String
Dim strMiddleProfileXScale As String
Dim strMiddleProfileYScale As String
Dim strTopProfileXScale As String
Dim strTopProfileYScale As String
Dim strRotationAngle As String
Dim strFloorToflootHeight As String
Dim strNumberOfLevel As String
If CheckBox_Massing_HKIFC.Checked Then
    strArchitecturalSyle = "Cesar_HKIFC"
End If
If CheckBox_Massing_Petronas.Checked Then
    strArchitecturalSyle = "Cesar_Petronas"
End If
If CheckBox_Cira.Checked Then
    strArchitecturalSyle = "Cesar_Cira"
End If
If CheckBox_HearstHeadquarter.Checked Then
    strArchitecturalSyle = "Massing_4"
End If
If CheckBox_Massing_Mini.Checked Then
    strArchitecturalSyle = "Mini_Skyscraper"
End If
If CheckBox_HearstHeadquarter.Checked Then
    strArchitecturalSyle = "Hearst_Headquarter"
End If
strMiddleProfileLevel = TrackBarMiddleProfileLevel.Value
strMiddleProfileXScale = TrackBarMiddleProfileXScale.Value
strMiddleProfileYScale = TrackBarMiddleProfileYScale.Value
strTopProfileXScale = TrackBarTopProfileXScale.Value
strTopProfileYScale = TrackBarTopProfileYScale.Value
strRotationAngle = TrackBarRotationAngle.Value
strFloorToflootHeight = TextBoxFloorHeight.Text
strNumberOfLevel = TextBoxFloorNumbers.Text
Dim strVerticalProportion As String
strVerticalProportion = TrackBarVerticalProportion.Value
If CheckBox_Massing_Derivative_Option.Checked = False Then
    RhUtil.RhinoApp.RunMenuScript("-LoadScript C:\thesis\Run\Digital-
Skyscraper_Massing_OriginalTowers.rvb " + strArchitecturalSyle + " " + strVer-
ticalProportion + " ")
End If
If CheckBox_Massing_Derivative_Option.Checked Then
    RhUtil.RhinoApp.RunMenuScript("-LoadScript C:\thesis\Run\Digital-
Skyscraper_Massing_DeviativeTowers.rvb " + strArchitecturalSyle + " " + strMid-
dleProfileLevel + " " + strMiddleProfileXScale + " " + strMiddleProfileYScale + "
" + strTopProfileXScale + " " + strTopProfileYScale + " " + strRotationAngle + "
" + strFloorToflootHeight + " " + strNumberOfLevel + " ")
End If
End Sub

```


Appendix B: Rhinoscript

This appendix contains the runtime program that was written in Rhinoscript. This program was tested fully functional by experiments. Please refer to Chapter xxx for the actual experiments that were conducted. (Only a small portion of the codes is shown. It is used to demonstrate the key ideas of the program.)

DigitalSkyscraper_Massing_OriginalTowers.RVB

```
Function Construct_HKIFC_Massing(architecturalStyle, pointA1, pointA2, pointA3, pointA4, tapered, verticalController)

Dim point1, point2, point3, point4
ReDim arrConstructPlanProfiles(10,1)
ReDim arrMassing(10)
point1 = pointA1
point2 = pointA2
point3 = pointA3
point4 = pointA4
Dim line11, line22, line33, line44
Dim outLine1(18)                                     `the final outline of the
plan
ReDim lineAllNew(3)                                  `array that contains four
lines
Dim int11, int22, int33, zLevel
Dim int111, int222, int333
Dim i, j
Dim veritcalLevel(10)
veritcalLevel(0) = 52*verticalController
veritcalLevel(1) = 65*verticalController
veritcalLevel(2) = 68*verticalController
veritcalLevel(3) = 77*verticalController
veritcalLevel(4) = 79*verticalController
veritcalLevel(5) = 84*verticalController
veritcalLevel(6) = 86*verticalController
veritcalLevel(7) = 88*verticalController
veritcalLevel(8) = 89*verticalController
veritcalLevel(9) = 90*verticalController

For i = 0 To 9
  ReDim zLevel(1)
  Select Case i
    Case 0
      int11 = 0
      int22 = 0
      int33 = 0
      zLevel(0) = 0
      zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(0), 89)
    Case 1
      int11 = 1
      int22 = 1
      int33 = 0
      zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(0), 89)
```

```

    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(1), 89)
Case 2
    int11 = 2
    int22 = 2
    int33 = 0
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(1), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(2), 89)
Case 3
    int11 = 2
    int22 = 2
    int33 = 1
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(2), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(3), 89)
Case 4
    int11 = 3
    int22 = 3
    int33 = 1
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(3), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(4), 89)
Case 5
    int11 = 3
    int22 = 3
    int33 = 2
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(4), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(5), 89)
Case 6
    int11 = 4
    int22 = 4
    int33 = 2
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(5), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(6), 89)
Case 7
    int11 = 4.8
    int22 = 4.8
    int33 = 3
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(6), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(7), 89)
Case 8
    int11 = 5
    int22 = 5
    int33 = 3
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(7), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(8), 89)
Case 9
    int11 = 0
    int22 = 0
    int33 = 4
    int111 = 0
    int222 = 0
    int333 = 6
    zLevel(0) = LevelFinder(architecturalStyle, veritcalLevel(7), 89)
    zLevel(1) = LevelFinder(architecturalStyle, veritcalLevel(9), 89)
End Select

If tapered = False Then
    If i<8 Then

```

```

    For j=0 To 1 Step 1
HKIFC_planProfile_3 point1, point2, point3, point4, int11, int22, int33,
zLevel(j)
line11 = Rhino.LastObject

HKIFC_planProfile_3 point2, point3, point4, point1, int11, int22, int33,
zLevel(j)
line22 = Rhino.LastObject

HKIFC_planProfile_3 point3, point4, point1, point2, int11, int22, int33,
zLevel(j)
line33 = Rhino.LastObject

HKIFC_planProfile_3 point4, point1, point2, point3, int11, int22, int33,
zLevel(j)
line44 = Rhino.LastObject

lineAllNew(0) = line11
lineAllNew(1) = line22
lineAllNew(2) = line33
lineAllNew(3) = line44

Rhino.JoinCurves lineAllNew
arrConstructPlanProfiles(i,j) = Rhino.LastObject
Rhino.DeleteObject line11
Rhino.DeleteObject line22
Rhino.DeleteObject line33
Rhino.DeleteObject line44
    Next

    ReDim aaa(1)
    Dim bbb, ccc
    aaa(0) = arrConstructPlanProfiles(i,0)
    aaa(1) = arrConstructPlanProfiles(i,1)
    If IsArray(aaa) And UBound(aaa) > 0 Then
Rhino.AddLoftSrf (aaa)
bbb = Rhino.LastObject
ccc = Rhino.SelectObject(bbb)
Rhino.Command ("_cap " & ccc & " ")
Rhino.UnselectAllObjects
    End If
    ElseIf i=9 Then

HKIFC_planProfile_2 point1, point2, point3, point4, int11, int22, int33, zLev-
el(0)
line11 = Rhino.LastObject

HKIFC_planProfile_2 point2, point3, point4, point1, int11, int22, int33, zLev-
el(0)
line22 = Rhino.LastObject

HKIFC_planProfile_2 point3, point4, point1, point2, int11, int22, int33, zLev-
el(0)
line33 = Rhino.LastObject

```

```

HKIFC_planProfile_2 point4, point1, point2, point3, int11, int22, int33, zLevel(0)
line44 = Rhino.LastObject

lineAllNew(0) = line11
lineAllNew(1) = line22
lineAllNew(2) = line33
lineAllNew(3) = line44

Rhino.JoinCurves lineAllNew
arrConstructPlanProfiles(i,0) = Rhino.LastObject
Rhino.DeleteObject line11
Rhino.DeleteObject line22
Rhino.DeleteObject line33
Rhino.DeleteObject line44

ReDim aaa(1)
Dim ddd, eee
aaa(0) = arrConstructPlanProfiles(i,0)

Dim line111, line222, line333, line444
Dim lineAllNew_2(3)
HKIFC_planProfile_2 point1, point2, point3, point4, int111, int222, int333,
zLevel(1)
line111 = Rhino.LastObject

HKIFC_planProfile_2 point2, point3, point4, point1, int111, int222, int333,
zLevel(1)
line222 = Rhino.LastObject

HKIFC_planProfile_2 point3, point4, point1, point2, int111, int222, int333,
zLevel(1)
line333 = Rhino.LastObject

HKIFC_planProfile_2 point4, point1, point2, point3, int111, int222, int333,
zLevel(1)
line444 = Rhino.LastObject

lineAllNew_2(0) = line111
lineAllNew_2(1) = line222
lineAllNew_2(2) = line333
lineAllNew_2(3) = line444

Rhino.JoinCurves lineAllNew_2
arrConstructPlanProfiles(i,1) = Rhino.LastObject
Rhino.DeleteObject line111
Rhino.DeleteObject line222
Rhino.DeleteObject line333
Rhino.DeleteObject line444

aaa(1) = arrConstructPlanProfiles(i,1)

If IsArray(aaa) And UBound(aaa) > 0 Then
Rhino.AddLoftSrf (aaa)
ddd = Rhino.LastObject
eee = Rhino.SelectObject(ddd)

```

```

Rhino.Command ("_cap " & eee & " ")
Rhino.UnselectAllObjects
End If
End If 'end of if i<8
Else
MsgBox("It is tapered!")
End If 'end of if tapered
Next
Construct_HKIFC_Massing = arrConstructPlanProfiles
End Function 'end of Construct_HKIFC_Massing

Function HKIFC_planProfile_1 (point1, point2, point3, point4, integerNumber1,
integerNumber2, integerNumber3, zLevel)
Dim int1, int2, int3
int1 = integerNumber1*1338.025
int2 = integerNumber2*1118.004
int3 = integerNumber3*1118.004
Dim z
z = zLevel
Dim point11, point22, point33, point44
point11 = point1
point22 = point2
point33 = point3
point44 = point4
Dim lineAngle1
lineAngle1 = Rhino.Angle(point11, point33)
Dim x1, y1, z1
Dim x2, y2, z2
Dim x3, y3, z3
Dim x4, y4, z4
x1 = point11(0)
y1 = point11(1)
z1 = point11(2)
x2 = point22(0)
y2 = point22(1)
z2 = point22(2)
x3 = point33(0)
y3 = point33(1)
z3 = point33(2)
x4 = point44(0)
y4 = point44(1)
z4 = point44(2)

'points that forms the outlines
Dim pt11, pt22, pt33, pt44, pt55, pt66, pt77, pt88, pt99
'reference points on alternative two sides
Dim xa, ya, za
Dim xb, yb, zb

'point pt11
x11 = (x3 - x1)*(6646.804+int1)/81175.858+x1
y11 = (y3 - y1)*(6646.804+int1)/81175.858+y1
z11 = (z3 - z1)*(6646.804+int1)/81175.858+z
pt11 = Array(x11, y11, z11)

```

```

`point pt22
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa2 = (x4 - x1)*(3591.415+int2)/57400+x1
ya2 = (y4 - y1)*3591.415/57400+y1
za2 = (z4 - z1)*3591.415/57400+z
xb2 = (x3 - x2)*(3591.415+int2)/57400+x2
yb2 = (y3 - y2)*3591.415/57400+y2
zb2 = (z3 - z2)*3591.415/57400+z
Else
xa2 = (x4 - x1)*3591.415/57400+x1
ya2 = (y4 - y1)*(3591.415+int2)/57400+y1
za2 = (z4 - z1)*3591.415/57400+z
xb2 = (x3 - x2)*3591.415/57400+x2
yb2 = (y3 - y2)*(3591.415+int2)/57400+y2
zb2 = (z3 - z2)*3591.415/57400+z
End If
x22 = (xb2 - xa2)*9963.189/57400+xa2
y22 = (yb2 - ya2)*9963.189/57400+ya2
z22 = (zb2 - za2)*9963.189/57400+za2
pt22 = Array(x22, y22, z22)

`point pt33
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa3 = (x4 - x1)*(2778+int2)/57400+x1
ya3 = (y4 - y1)*2778/57400+y1
za3 = (z4 - z1)*2778/57400+z
xb3 = (x3 - x2)*(2778+int2)/57400+x2
yb3 = (y3 - y2)*2778/57400+y2
zb3 = (z3 - z2)*2778/57400+z
Else
xa3 = (x4 - x1)*2778/57400+x1
ya3 = (y4 - y1)*(2778+int2)/57400+y1
za3 = (z4 - z1)*2778/57400+z
xb3 = (x3 - x2)*2778/57400+x2
yb3 = (y3 - y2)*(2778+int2)/57400+y2
zb3 = (z3 - z2)*2778/57400+z
End If
x33 = (xb3 - xa3)*15280/57400+xa3
y33 = (yb3 - ya3)*15280/57400+ya3
z33 = (zb3 - za3)*15280/57400+za3
pt33 = Array(x33, y33, z33)

`point pt44
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa4 = (x4 - x1)*(900+int3)/57400+x1
ya4 = (y4 - y1)*900/57400+y1
za4 = (z4 - z1)*900/57400+z
xb4 = (x3 - x2)*(900+int3)/57400+x2
yb4 = (y3 - y2)*900/57400+y2
zb4 = (z3 - z2)*900/57400+z
Else
xa4 = (x4 - x1)*900/57400+x1
ya4 = (y4 - y1)*(900+int3)/57400+y1
za4 = (z4 - z1)*900/57400+z
xb4 = (x3 - x2)*900/57400+x2

```

```

yb4 = (y3 - y2)*(900+int3)/57400+y2
zb4 = (z3 - z2)*900/57400+z
End If
x44 = (xb4 - xa4)*15280/57400+xa4
y44 = (yb4 - ya4)*15280/57400+ya4
z44 = (zb4 - za4)*15280/57400+za4
pt44 = Array(x44, y44, z44)

'point pt55
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa5 = (x4 - x1)*(0+int3)/57400+x1
ya5 = (y4 - y1)*0/57400+y1
za5 = (z4 - z1)*0/57400+z
xb5 = (x3 - x2)*(0+int3)/57400+x2
yb5 = (y3 - y2)*0/57400+y2
zb5 = (z3 - z2)*0/57400+z
Else
xa5 = (x4 - x1)*0/57400+x1
ya5 = (y4 - y1)*(0+int3)/57400+y1
za5 = (z4 - z1)*0/57400+z
xb5 = (x3 - x2)*0/57400+x2
yb5 = (y3 - y2)*(0+int3)/57400+y2
zb5 = (z3 - z2)*0/57400+z
End If
x55 = (xb5 - xa5)*28700/57400+xa5
y55 = (yb5 - ya5)*28700/57400+ya5
z55 = (zb5 - za5)*28700/57400+za5
pt55 = Array(x55, y55, z55)

'point pt66
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa6 = (x4 - x1)*(900+int3)/57400+x1
ya6 = (y4 - y1)*900/57400+y1
za6 = (z4 - z1)*900/57400+z
xb6 = (x3 - x2)*(900+int3)/57400+x2
yb6 = (y3 - y2)*900/57400+y2
zb6 = (z3 - z2)*900/57400+z
Else
xa6 = (x4 - x1)*900/57400+x1
ya6 = (y4 - y1)*(900+int3)/57400+y1
za6 = (z4 - z1)*900/57400+z
xb6 = (x3 - x2)*900/57400+x2
yb6 = (y3 - y2)*(900+int3)/57400+y2
zb6 = (z3 - z2)*900/57400+z
End If
x66 = (xb6 - xa6)*42120/57400+xa6
y66 = (yb6 - ya6)*42120/57400+ya6
z66 = (zb6 - za6)*42120/57400+za6
pt66 = Array(x66, y66, z66)

'point pt77
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa7 = (x4 - x1)*(2778+int2)/57400+x1
ya7 = (y4 - y1)*2778/57400+y1
za7 = (z4 - z1)*2778/57400+z

```

```

xb7 = (x3 - x2)*(2778+int2)/57400+x2
yb7 = (y3 - y2)*2778/57400+y2
zb7 = (z3 - z2)*2778/57400+z
Else
xa7 = (x4 - x1)*2778/57400+x1
ya7 = (y4 - y1)*(2778+int2)/57400+y1
za7 = (z4 - z1)*2778/57400+z
xb7 = (x3 - x2)*2778/57400+x2
yb7 = (y3 - y2)*(2778+int2)/57400+y2
zb7 = (z3 - z2)*2778/57400+z
End If
x77 = (xb7 - xa7)*42120/57400+xa7
y77 = (yb7 - ya7)*42120/57400+ya7
z77 = (zb7 - za7)*42120/57400+za7
pt77 = Array(x77, y77, z77)

'point pt88
If lineAngle1(0)>0 And lineAngle1(0)<90 Or lineAngle1(0)<-90 Then
xa8 = (x4 - x1)*(3739+int2)/57400+x1
ya8 = (y4 - y1)*3739/57400+y1
za8 = (z4 - z1)*3739/57400+z
xb8 = (x3 - x2)*(3739+int2)/57400+x2
yb8 = (y3 - y2)*3739/57400+y2
zb8 = (z3 - z2)*3739/57400+z
Else
xa8 = (x4 - x1)*3739/57400+x1
ya8 = (y4 - y1)*(3739+int2)/57400+y1
za8 = (z4 - z1)*3739/57400+z
xb8 = (x3 - x2)*3739/57400+x2
yb8 = (y3 - y2)*(3739+int2)/57400+y2
zb8 = (z3 - z2)*3739/57400+z
End If
x88 = (xb8 - xa8)*47410/57400+xa8
y88 = (yb8 - ya8)*47410/57400+ya8
z88 = (zb8 - za8)*47410/57400+za8
pt88 = Array(x88, y88, z88)

'point pt99
x99 = (x4 - x2)*(6646.804+int1)/81175.858+x2
y99 = (y4 - y2)*(6646.804+int1)/81175.858+y2
z99 = (z4 - z2)*6646.804/81175.858+z
pt99 = Array(x99, y99, z99)

'outlines that compose one size
Dim line1, line2, line3, line4, line5, line55

line1 = Rhino.AddArc3Pt (pt11, pt33, pt22)
line2 = Rhino.AddLine (pt33, pt44)
line3 = Rhino.AddArc3Pt (pt44, pt66, pt55)
line4 = Rhino.AddLine (pt66, pt77)
line5 = Rhino.AddArc3Pt (pt77, pt99, pt88)
Rhino.SelectObject line5
Rhino.Command("-rebuild " & "" & "Enter")
line55 = Rhino.LastObject

Dim lineAll(4)

```



```

lineAll(0) = line1
lineAll(1) = line2
lineAll(2) = line3
lineAll(3) = line4
lineAll(4) = line55
HKIFC_planProfile_1 = Rhino.JoinCurves (lineAll)

Rhino.DeleteObject line5
Rhino.DeleteObject line55
Rhino.DeleteObject line4
Rhino.DeleteObject line3
Rhino.DeleteObject line2
Rhino.DeleteObject line1
End Function                                     `end of HKIFC_planProfile_1

Function LevelFinder(architecturalStyle, i, numberOfLevels)

Select Case architecturalStyle
  Case "Mini_Skyscraper"
    LevelFinder = i * 4200

  Case "Cesar_HKIFC"
    Dim countLevel
    countLevel=Array(5240,5000,5000,5410,7750,7050,7050,4200,4850,4200,4200,48
50,4200,4200,4850,4200,4200,4850, 4200,4200,4850,4200,4200,4850,4200,4200,485
0,4420,3400,6880,6880,4200,4850,4200,4200,4200,4200,4200, 4200,4200,4200,4200
,4200,4200,4200,4200,4200,4200,4420,3400,6880,6880,5240,5130,4850,4200,4850,4
200,4200, 4850,4200,4850,4200,4850,4420,7030,7030,3400,6880,6880,4200,4850,42
00,4200,4850,4200,4200,4200,4850,4200,4850, 4200,4200,4200,4850,4850,4200,420
0,4850,4350)
    If i=0 Then
      LevelFinder=5240
    Else
      LevelFinder=5240
      For j=i To 1 Step -1
        LevelFinder = LevelFinder + countLevel(j)
      Next
    End If

    If i>5 And i<30 Then
      LevelFinder = LevelFinder-7050
    End If
    If i>29 And i<51 Then
      LevelFinder = LevelFinder-7050-6880
    End If
    If i>50 And i<66 Then
      LevelFinder = LevelFinder-7050-6880-6880
    End If
    If i>65 And i<69 Then
      LevelFinder = LevelFinder-7050-6880-6880-7030
    End If
    If i>68 Then
      LevelFinder = LevelFinder-7050-6880-6880-7030-6880
    End If

```

```

Case "Cesar_Petronas"
  Dim countLevel2
  countLevel2 = Array(5500,5500,5500,5500,5500, 4100,4100,4100,4100,
4100,4100,4100,4100, 4100,4100,4100,4100, 4100,4100,4100,4100,
4100,4100,4100,4100, _
4100,4100,4100,4100, 4100,4100,4100,4100, 4100,4100,4100, 4505,3900,3900,43
50,4100,4100,4350, 4100, 4100,4100,4100,4100,4100,4100,4100,4100,4100,4100,
4100,4100,4100,4100)

  If i=0 Then
    LevelFinder=5500
  Else
    LevelFinder=5500
    For j=i To 1 Step -1
      LevelFinder = LevelFinder + countLevel2(j)
    Next
  End If
Case Else
  Rhino.MessageBox ("something else!")
End Select
End Function      `end of LevelFinder

```

DigitalSkyscraper_Massing_DerivativeTowers.RVB

```

Function MassingController (point1, point2, point3, point4, architectural-
Style, middleProfileLevel, middleProfileXScale, middleProfileYScale, topProfileX-
Scale, topProfileYScale, rotationAngle, floorTofloatHeight, numberOfLevel)

```

```

Dim line11, line22, line33, line44
ReDim lineAllNew(3)
Dim lowerProfile, middleProfile, topProfile
Dim lowerProfile2, middleProfile2, topProfile2
Dim lowerProfile3, middleProfile3, topProfile3
Dim reVerticalLevelSlider
reVerticalLevelSlider = VerticalLevelSlider (middleProfileLevel)
Dim reHorizontalSizeSlider
reHorizontalSizeSlider = VerticalLevelSlider (2)
Dim intersectOfPt1Pt2Pt3Pt4
Dim interPoint
FindIntersectionPoints point1, point2, point3, point4
interPoint = Rhino.LastObject
intersectOfPt1Pt2Pt3Pt4 = Rhino.PointCoordinates (interPoint)
ReDim intersectOfMiddelLevel(2)
ReDim intersectOfTopLevel(2)
intersectOfMiddelLevel(0) = intersectOfPt1Pt2Pt3Pt4(0)
intersectOfMiddelLevel(1) = intersectOfPt1Pt2Pt3Pt4(1)
intersectOfMiddelLevel(2) = floorTofloatHeight*numberOfLevel*reVerticalLevelSli
der

intersectOfTopLevel(0) = intersectOfPt1Pt2Pt3Pt4(0)
intersectOfTopLevel(1) = intersectOfPt1Pt2Pt3Pt4(1)
intersectOfTopLevel(2) = floorTofloatHeight*numberOfLevel

Rhino.Print (CStr(intersectOfMiddelLevel(0)))

```

```

Rhino.Print (CStr(intersectOfMiddelLevel(1)))
Rhino.Print (CStr(intersectOfMiddelLevel(2)))
Rhino.Print (CStr(intersectOfTopLevel(0)))
Rhino.Print (CStr(intersectOfTopLevel(1)))
Rhino.Print (CStr(intersectOfTopLevel(2)))

If architecturalStyle = "Cesar_HKIFC" Then
`lower profile
HKIFC_planProfile_3 point1, point2, point3, point4, 0, 0, 0, 0
line11 = Rhino.LastObject

HKIFC_planProfile_3 point2, point3, point4, point1, 0, 0, 0, 0
line22 = Rhino.LastObject

HKIFC_planProfile_3 point3, point4, point1, point2, 0, 0, 0, 0
line33 = Rhino.LastObject

HKIFC_planProfile_3 point4, point1, point2, point3, 0, 0, 0, 0
line44 = Rhino.LastObject

lineAllNew(0) = line11
lineAllNew(1) = line22
lineAllNew(2) = line33
lineAllNew(3) = line44
Rhino.JoinCurves lineAllNew
lowerProfile = Rhino.LastObject
Rhino.DeleteObject line11
Rhino.DeleteObject line22
Rhino.DeleteObject line33
Rhino.DeleteObject line44

`middel profile
HKIFC_planProfile_3 point1, point2, point3, point4, 0, 0, 0, floorTofloatHeight*
numberOfLevel*reVerticalLevelSlider
line11 = Rhino.LastObject

HKIFC_planProfile_3 point2, point3, point4, point1, 0, 0, 0, floorTofloatHeight*
numberOfLevel*reVerticalLevelSlider
line22 = Rhino.LastObject

HKIFC_planProfile_3 point3, point4, point1, point2, 0, 0, 0, floorTofloatHeight*
numberOfLevel*reVerticalLevelSlider
line33 = Rhino.LastObject

HKIFC_planProfile_3 point4, point1, point2, point3, 0, 0, 0, floorTofloatHeight*
numberOfLevel*reVerticalLevelSlider
line44 = Rhino.LastObject

lineAllNew(0) = line11
lineAllNew(1) = line22
lineAllNew(2) = line33
lineAllNew(3) = line44
Rhino.JoinCurves lineAllNew
middleProfile2 = Rhino.LastObject
Rhino.DeleteObject line11

```

```

Rhino.DeleteObject line22
Rhino.DeleteObject line33
Rhino.DeleteObject line44
Dim numb1, numb2
numb1 = middleProfileXScale
numb2 = middleProfileYScale
Dim arrScaleM
arrScaleM = Array(numb1, numb2, 1)
Rhino.ScaleObject middleProfile2, intersectOfMiddleLevel, arrScaleM, True
middleProfile3 = Rhino.LastObject
Rhino.DeleteObject middleProfile2
Rhino.RotateObject middleProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle/2,
, True
middleProfile = Rhino.LastObject
Rhino.DeleteObject middleProfile3

`top profile
HKIFC_planProfile_3 point1, point2, point3, point4, 0, 0, 0, floorTofloatHeight*
numberOfLevel
line11 = Rhino.LastObject

HKIFC_planProfile_3 point2, point3, point4, point1, 0, 0, 0, floorTofloatHeight*
numberOfLevel
line22 = Rhino.LastObject

HKIFC_planProfile_3 point3, point4, point1, point2, 0, 0, 0, floorTofloatHeight*
numberOfLevel
line33 = Rhino.LastObject

HKIFC_planProfile_3 point4, point1, point2, point3, 0, 0, 0, floorTofloatHeight*
numberOfLevel
line44 = Rhino.LastObject

lineAllNew(0) = line11
lineAllNew(1) = line22
lineAllNew(2) = line33
lineAllNew(3) = line44
Rhino.JoinCurves lineAllNew
topProfile2 = Rhino.LastObject
Rhino.DeleteObject line11
Rhino.DeleteObject line22
Rhino.DeleteObject line33
Rhino.DeleteObject line44
Dim arrScaleT
numb1 = topProfileXScale
numb2 = topProfileYScale
arrScaleT = Array(numb1, numb2, 1)
Rhino.ScaleObject topProfile2, intersectOfTopLevel, arrScaleT, True
Rhino.DeleteObject topProfile2
topProfile3 = Rhino.LastObject
Rhino.RotateObject topProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle, , True
topProfile = Rhino.LastObject
Rhino.DeleteObject topProfile3

ReDim aaa(2)

```

```

Dim bbb, ccc
aaa(0) = lowerProfile
aaa(1) = middleProfile
aaa(2) = topProfile
If IsArray(aaa) And UBound(aaa) > 0 Then
Rhino.AddLoftSrf (aaa)
bbb = Rhino.LastObject
ccc = Rhino.SelectObject(bbb)
Rhino.Command ("_cap " & ccc & " ")
Rhino.UnselectAllObjects
End If `end of If IsArray
End If `end of If "Cesar_HKIFC"
If architecturalStyle = "Mini_Skyscraper" Then

`lower profile
Mini_planProfile point1, point2, point3, point4, 0
lowerProfile = Rhino.LastObject

`middel profile
Mini_planProfile point1, point2, point3, point4, floorTofloatHeight*numberOfLeve
l*reVerticalLevelSlider
middleProfile2 = Rhino.LastObject
`Dim numb1, numb2
numb1 = middleProfileXScale
numb2 = middleProfileYScale
`Dim arrScaleM
arrScaleM = Array(numb1, numb2, 1)
Rhino.ScaleObject middleProfile2, intersectOfMiddelLevel, arrScaleM, True
middleProfile3 = Rhino.LastObject
Rhino.DeleteObject middleProfile2
Rhino.RotateObject middleProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle/2,
,True
middleProfile = Rhino.LastObject
Rhino.DeleteObject middleProfile3

`top profile
Mini_planProfile point1, point2, point3, point4, floorTofloatHeight*numberOfLeve
l
topProfile2 = Rhino.LastObject

numb1 = topProfileXScale
numb2 = topProfileYScale
arrScaleT = Array(numb1, numb2, 1)
Rhino.ScaleObject topProfile2, intersectOfTopLevel, arrScaleT, True
Rhino.DeleteObject topProfile2
topProfile3 = Rhino.LastObject
Rhino.RotateObject topProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle, ,True
topProfile = Rhino.LastObject
Rhino.DeleteObject topProfile3

ReDim aaa(2)
aaa(0) = lowerProfile
aaa(1) = middleProfile
aaa(2) = topProfile
If IsArray(aaa) And UBound(aaa) > 0 Then

```

```

Rhino.AddLoftSrf (aaa)
bbb = Rhino.LastObject
ccc = Rhino.SelectObject(bbb)
Rhino.Command ("_cap " & ccc & " ")
Rhino.UnselectAllObjects
End If 'end of If IsArray
End If 'end of If "Mini_Skyscraper"

If architecturalStyle = "Cesar_Petronas" Then
`lower profile
Petronas_planProfile point1, point2, point3, point4, 0
lowerProfile = Rhino.LastObject

`middel profile
Petronas_planProfile point1, point2, point3, point4, floorToflootHeight*numberOfLe
vel*reVerticalLevelSlider
middleProfile2 = Rhino.LastObject
`Dim numb1, numb2
numb1 = middleProfileXScale
numb2 = middleProfileYScale
arrScaleM = Array(numb1, numb2, 1)
Rhino.ScaleObject middleProfile2, intersectOfMiddelLevel, arrScaleM, True
middleProfile3 = Rhino.LastObject
Rhino.DeleteObject middleProfile2
Rhino.RotateObject middleProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle/2,
,True
middleProfile = Rhino.LastObject
Rhino.DeleteObject middleProfile3

`top profile
Petronas_planProfile point1, point2, point3, point4, floorToflootHeight*numberOfLe
vel
topProfile2 = Rhino.LastObject

numb1 = topProfileXScale
numb2 = topProfileYScale
arrScaleT = Array(numb1, numb2, 1)
Rhino.ScaleObject topProfile2, intersectOfTopLevel, arrScaleT, True
Rhino.DeleteObject topProfile2
topProfile3 = Rhino.LastObject
Rhino.RotateObject topProfile3, intersectOfPt1Pt2Pt3Pt4, rotationAngle, ,True
topProfile = Rhino.LastObject
Rhino.DeleteObject topProfile3

ReDim aaa(2)
aaa(0) = lowerProfile
aaa(1) = middleProfile
aaa(2) = topProfile
If IsArray(aaa) And UBound(aaa) > 0 Then
Rhino.AddLoftSrf (aaa)
bbb = Rhino.LastObject
ccc = Rhino.SelectObject(bbb)
Rhino.Command ("_cap " & ccc & " ")
Rhino.UnselectAllObjects
End If 'end of If IsArray

```

```

End If 'end of If "Cesar_Petronas"
End If

End Function

Sub HKIFC_componentConstruct(architecturalStyle, numberOfHorizontalMullions,
horizontalMullionsStyle, verticalMullionWidth, verticalMullionLength, hori-
zontalMullionWidth, horizontalMullionHeight, horizontalLouvreSize_Width, com-
ponentHorizontalSpan, verticalMullionWidthTapperRadio, startLevel, endLevel)

Dim numberOfSurfaces
Dim innerPoint
innerPoint = Array(0,0,0)
Dim curvesAll
Dim numofDivisions
Dim distance1
Dim i, j, r, rt, srfCount
Dim diDiagridPointPositions
Dim numofLevels

Dim selectedSurfaces
selectedSurfaces = Rhino.GetObjects("Pick surface objects")
numberOfSurfaces = UBound(selectedSurfaces)+1

Dim lowestPoint, highestPoint
lowestPoint = Rhino.GetPoint("Pick the lowestPoint on the surfaces:")
highestPoint = Rhino.GetPoint("Pick the highestPoint on the surfaces:")
Dim heightOfSelectedSurface
heightOfSelectedSurface = highestPoint(2)
Dim lowestPointZCoordinate
lowestPointZCoordinate = lowestPoint(2)
,
'determines the number of division
,

Dim ptpt1, ptpt2, ptpt3, ptpt4
ptpt1=Array(-100000,-100000,lowestPointZCoordinate)
ptpt2=Array(100000,100000,lowestPointZCoordinate)
Rhino.Command "_cplane " & "w" & " " & "t" & " "
Rhino.Command "_Plane " & Pt2Str(ptpt1) & " " & Pt2Str(ptpt2)
Dim cutPlaneTest
cutPlaneTest=Rhino.LastObject
Rhino.SelectObject(selectedSurfaces(0))
Rhino.SelectObject(cutPlaneTest)
Rhino.Command " Intersect "
Dim intersectedLinesTest
intersectedLinesTest=Rhino.LastObject
Rhino.DeleteObject cutPlaneTest
Rhino.UnselectAllObjects
Dim interCurveLength
interCurveLength = Rhino.CurveLength(intersectedLinesTest)
numofDivisions = CInt(interCurveLength/componentHorizontalSpan)

numofLevels = heightOfSelectedSurface/4200
Dim countStartLevel, countEndLevel

```

```

If endLevel=100 Then
countStartLevel = 0
countEndLevel = numofLevels-1
Else
countStartLevel = startLevel
countEndLevel = endLevel
End If
Dim pointCloud11(1000000)
Dim pointCloud33(1000000)

For i=countStartLevel To countEndLevel `for i
For srfCount=0 To numberOfSurfaces-1 `for srfCount
Dim cutPlane, cutPlane2
r = LevelFinder(architecturalSytle, i, numofLevels)
Dim pt1, pt2, pt3, pt4
pt1=Array(-100000,-100000,r)
pt2=Array(100000,100000,r)
Rhino.Command "_cplane " & "w" & " " & "t" & " "
Rhino.Command "_Plane " & Pt2Str(pt1) & " " & Pt2Str(pt2)
cutPlane=Rhino.LastObject
Rhino.SelectObject(selectedSurfaces(srfCount))
Rhino.SelectObject(cutPlane)
Rhino.Command "_Intersect "
intersectedLines=Rhino.LastObject
Rhino.DeleteObject cutPlane
Rhino.UnselectAllObjects

pointCloud11(srfCount) = FindDivisionPoints(intersectedLines, numofDivi-
sions)
rt = LevelFinder(architecturalSytle, i+1, numofLevels)
pt3=Array(-100000,-100000,rt)
pt4=Array(100000,100000,rt)
Rhino.Command "_cplane " & "w" & " " & "t" & " "
Rhino.Command "_Plane " & Pt2Str(pt3) & " " & Pt2Str(pt4)
cutPlane2=Rhino.LastObject
Rhino.SelectObject(selectedSurfaces(srfCount))
Rhino.SelectObject(cutPlane2)
Rhino.Command "_Intersect "
intersectedLines2=Rhino.LastObject
Rhino.DeleteObject cutPlane2
Rhino.UnselectAllObjects

pointCloud33(srfCount) = FindDivisionPoints(intersectedLines2, numofDivi-
sions)
Next `of srfCount

For srfCount=0 To numberOfSurfaces-1 `for srfCount
For j=0 To numofDivisions-1
If srfCount=0 Then
point1 = pointCloud11(srfCount)(j)
point2 = pointCloud33(srfCount)(j)
point3 = pointCloud11(srfCount)(j+1)
point4 = pointCloud33(srfCount)(j+1)

If j=0 Then

```



```

If CLng(pointCloud11(0)(0)(0)/100) = CLng(pointCloud11(numberOfSurfaces-
1)(numOfDivisions)(0)/100) Or _
CLng(pointCloud11(0)(0)(1)/100) = CLng(pointCloud11(numberOfSurfaces-1)(nu-
mOfDivisions)(1)/100) And _
CLng(pointCloud11(0)(0)(2)/100) = CLng(pointCloud11(numberOfSurfaces-1)(nu-
mOfDivisions)(2)/100) Then
point5 = pointCloud11(numberOfSurfaces-1)(numOfDivisions-1)
point6 = pointCloud33(numberOfSurfaces-1)(numOfDivisions-1)
Else
point5 = Array(2*point1(0)-point3(0), 2*point1(1)-point3(1), 2*point1(2)-
point3(2))
point6 = Array(2*point2(0)-point4(0), 2*point2(1)-point4(1), 2*point2(2)-
point4(2))
End If `end of if pointCloud11(srfCount,0) = pointCloud11(srfCount,numOfDivis-
ions-1)
End If `end of j=0

If j=numOfDivisions-1 Then
If numberOfSurfaces>1 Then
If CLng(pointCloud11(srfCount+1)(0)(0)/100) = CLng(pointCloud11(0)(numOfDivis-
ions)(0)/100) Or _
CLng(pointCloud11(srfCount+1)(0)(1)/100) = CLng(pointCloud11(0)(numOfDivision-
s)(1)/100) And _
CLng(pointCloud11(srfCount+1)(0)(2)/100) = CLng(pointCloud11(0)(numOfDivision-
s)(2)/100) Then
point7 = pointCloud11(srfCount+1)(1)
point8 = pointCloud33(srfCount+1)(1)
Else
point7 = Array(2*point3(0)-point1(0), 2*point3(1)-point1(1), 2*point3(2)-
point1(2))
point8 = Array(2*point4(0)-point2(0), 2*point4(1)-point2(1), 2*point4(2)-
point2(2))
End If `end of if pointCloud11(numberOfSurfaces-1,numOfDivisions-1) = point-
Cloud11(0,0)
Else
point7 = Array(2*point3(0)-point1(0), 2*point3(1)-point1(1), 2*point3(2)-
point1(2))
point8 = Array(2*point4(0)-point2(0), 2*point4(1)-point2(1), 2*point4(2)-
point2(2))
End If `numberOfSurfaces>1
End If `j=numOfDivisions-1

If j>0 Then
point5 = pointCloud11(srfCount)(j-1)
point6 = pointCloud33(srfCount)(j-1)
End If

If j<numOfDivisions-1 Then
point7 = pointCloud11(srfCount)(j+2)
point8 = pointCloud33(srfCount)(j+2)
End If

ElseIf srfCount=numberOfSurfaces-1 Then

point1 = pointCloud11(srfCount)(j)

```

```

point2 = pointCloud33(srfCount)(j)
point3 = pointCloud11(srfCount)(j+1)
point4 = pointCloud33(srfCount)(j+1)

If j= 0 Then
If CLng(pointCloud11(numberOfSurfaces-1)(0)(0)/100) = CLng(pointCloud11(numberOfSurfaces-2)(numOfDivisions)(0)/100) Or _
CLng(pointCloud11(numberOfSurfaces-1)(0)(1)/100) = CLng(pointCloud11(numberOfSurfaces-2)(numOfDivisions)(1)/100) And _
CLng(pointCloud11(numberOfSurfaces-1)(0)(2)/100) = CLng(pointCloud11(numberOfSurfaces-2)(numOfDivisions)(2)/100) Then
point5 = pointCloud11(numberOfSurfaces-2)(numOfDivisions-1)
point6 = pointCloud33(numberOfSurfaces-2)(numOfDivisions-1)
Else
point5 = Array(2*point1(0)-point3(0), 2*point1(1)-point3(1), 2*point1(2)-point3(2))
point6 = Array(2*point2(0)-point4(0), 2*point2(1)-point4(1), 2*point2(2)-point4(2))
End If `end of if pointCloud11(numberOfSurfaces-1,numOfDivisions-1) = pointCloud11(0,0)
End If `j=numOfDivisions-1

If j=numOfDivisions-1 Then
If CLng(pointCloud11(numberOfSurfaces-1)(numOfDivisions)(0)/100) = CLng(pointCloud11(0)(0)(0)/100) Or _
CLng(pointCloud11(numberOfSurfaces-1)(numOfDivisions)(1)/100) = CLng(pointCloud11(0)(0)(1)/100) And _
CLng(pointCloud11(numberOfSurfaces-1)(numOfDivisions)(2)/100) = CLng(pointCloud11(0)(0)(2)/100) Then
point7 = pointCloud11(0)(1)
point8 = pointCloud33(0)(1)
Else
point7 = Array(2*point3(0)-point1(0), 2*point3(1)-point1(1), 2*point3(2)-point1(2))
point8 = Array(2*point4(0)-point2(0), 2*point4(1)-point2(1), 2*point4(2)-point2(2))
End If `end of if pointCloud11(srfCount,0) = pointCloud11(srfCount,numOfDivisions-1)
End If `end of j=0

If j>0 Then
point5 = pointCloud11(srfCount)(j-1)
point6 = pointCloud33(srfCount)(j-1)
End If
If j<numOfDivisions-1 Then
point7 = pointCloud11(srfCount)(j+2)
point8 = pointCloud33(srfCount)(j+2)
End If
Else
point1 = pointCloud11(srfCount)(j)
point2 = pointCloud33(srfCount)(j)
point3 = pointCloud11(srfCount)(j+1)
point4 = pointCloud33(srfCount)(j+1)

If j=0 Then

```

```

If CLng(pointCloud11(srfCount)(0)(0)/100) = CLng(pointCloud11(srfCount-1)(numOfDivisions)(0)/100) Or _
CLng(pointCloud11(srfCount)(0)(1)/100) = CLng(pointCloud11(srfCount-1)(numOfDivisions)(1)/100) And _
CLng(pointCloud11(srfCount)(0)(2)/100) = CLng(pointCloud11(srfCount-1)(numOfDivisions)(2)/100) Then
point5 = pointCloud11(srfCount-1)(numOfDivisions-1)
point6 = pointCloud33(srfCount-1)(numOfDivisions-1)
Else
point5 = Array(2*point1(0)-point3(0), 2*point1(1)-point3(1), 2*point1(2)-point3(2))
point6 = Array(2*point2(0)-point4(0), 2*point2(1)-point4(1), 2*point2(2)-point4(2))
End If 'end of if pointCloud11(srfCount,0) = pointCloud11(srfCount,numOfDivisions-1)
End If 'end of j=0

If j=numOfDivisions-1 Then
If CLng(pointCloud11(srfCount)(numOfDivisions)(0)/100) = CLng(pointCloud11(srfCount+1)(0)(0)/100) Or _
CLng(pointCloud11(srfCount)(numOfDivisions)(1)/100) = CLng(pointCloud11(srfCount+1)(0)(1)/100) And _
CLng(pointCloud11(srfCount)(numOfDivisions)(2)/100) = CLng(pointCloud11(srfCount+1)(0)(2)/100) Then
point7 = pointCloud11(srfCount+1)(1)
point8 = pointCloud33(srfCount+1)(1)
Else
point7 = Array(2*point3(0)-point1(0), 2*point3(1)-point1(1), 2*point3(2)-point1(2))
point8 = Array(2*point4(0)-point2(0), 2*point4(1)-point2(1), 2*point4(2)-point2(2))
End If 'end of if pointCloud11(numberOfSurfaces-1,numOfDivisions-1) = pointCloud11(0,0)
End If 'j=numOfDivisions-1

If j>0 Then
point5 = pointCloud11(srfCount)(j-1)
point6 = pointCloud33(srfCount)(j-1)
End If
If j<numOfDivisions-1 Then
point7 = pointCloud11(srfCount)(j+2)
point8 = pointCloud33(srfCount)(j+2)
End If
End If 'end of if srfCount=0

Dim var_VeriticalMullionWidthChangingValue
var_VeriticalMullionWidthChangingValue = verticalMullionWidth - i*5*verticalMullionWidthTapperRadio
Dim var_VeriticalMullionLengthChangingValue
var_VeriticalMullionLengthChangingValue = verticalMullionLength - i*6.5*verticalMullionWidthTapperRadio

'run Making function
If j=0 Then
HKIFC_component_2 architecturalSytle, numberOfHorizontalMullions, horizontal-

```

```

MullionsStyle, var_VerticalMullionWidthChangingValue, var_VerticalMullion-
LengthChangingValue, horizontalMullionWidth, horizontalMullionHeight, hori-
zontalLouvreSize_Width, innerPoint, point1, point2, point3, point4, point5,
point6, point7, point8, verticalMullionWidthTapperRadio, True, False
ElseIf j=numOfDivisions-1 Then
HKIFC_component_2 architecturalSytle, numberOfHorizontalMullions, horizontal-
MullionsStyle, var_VerticalMullionWidthChangingValue, var_VerticalMullion-
LengthChangingValue, horizontalMullionWidth, horizontalMullionHeight, hori-
zontalLouvreSize_Width, innerPoint, point1, point2, point3, point4, point5,
point6, point7, point8, verticalMullionWidthTapperRadio, False, True
ElseIf i=30 Or i=60 Or i=90 Then
HKIFC_component_3 architecturalSytle, numberOfHorizontalMullions, horizontal-
MullionsStyle, var_VerticalMullionWidthChangingValue, var_VerticalMullion-
LengthChangingValue, horizontalMullionWidth, horizontalMullionHeight, hori-
zontalLouvreSize_Width, innerPoint, point1, point2, point3, point4, point5,
point6, point7, point8, verticalMullionWidthTapperRadio
Else
HKIFC_component_3 architecturalSytle, numberOfHorizontalMullions, horizon-
talMullionsStyle, var_VerticalMullionWidthChangingValue, var_VerticalMullion-
LengthChangingValue, horizontalMullionWidth, horizontalMullionHeight, hori-
zontalLouvreSize_Width, innerPoint, point1, point2, point3, point4, point5,
point6, point7, point8, verticalMullionWidthTapperRadio
End If
Next 'of j span
Next 'for srfCount
Next 'of i level
End Sub 'end of sub HKIFC_componentConstruct

```

```

Sub HKIFC_component_1(architecturalStyle, numberOfHorizontalMullions, hori-
zontalMullionsStyle, verticalMullionWidth, verticalMullionLength, horizon-
talMullionWidth, horizontalMullionHeight, horizontalLouvreSize_Width, in-
nerPoint, pointm1, pointm2, pointm3, pointm4, pointm5, pointm6, pointm7,
pointm8, verticalMullionWidthTapperRadio)

```

```

'Vertical parametric
Dim var_floorHeight 'not the real floor height
Dim var_numberOfHorizontalMullions
Dim var_horizontalMullionsStyle
Dim var_distanceToLowerPoint
var_numberOfHorizontalMullions = numberOfHorizontalMullions
var_horizontalMullionsStyle = horizontalMullionsStyle

'Horizontal parametric
Dim distanceBetweenTopTwoPoints
Dim distanceBetweenButtomTwoPoints
Dim distanceBetweenPoint1Point2 'to measure the floor height

'variables for horizontal mullions
Dim var_HorizontalMullionWidth
Dim var_HorizontalMullionHeight
var_HorizontalMullionWidth = horizontalMullionWidth
var_HorizontalMullionHeight = horizontalMullionHeight

'variables for vertical mullions

```

```

Dim var_VerticalMullionWidth
Dim var_VerticalMullionLength
var_VerticalMullionWidth = verticalMullionWidth
var_VerticalMullionLength = verticalMullionLength

`variable for horizontal louvre
Dim var_HorizontalLouvreSize_Width
var_HorizontalLouvreSize_Width = horizontalLouvreSize_Width
`declare 8 points for construction, from point1 to point4 are
picked from users input
`point5 to point8 are reference points that positioned by the pro-
gram itself.
Dim point1, point2, point3, point4, point5, point6, point7(2), point8(2),
point9(2), point10(2), point11(2), point12(2), point13(2), point14(2)
`associated points
Dim point111, point222, point444, point555, point333, point666
`reference points
Dim refPoint5, refPoint6, refPoint7, refPoint8
point1 = pointm1
point2 = pointm2
point4 = pointm3
point5 = pointm4
point111 = pointm5
point222 = pointm6
point444 = pointm7
point555 = pointm8
point3 = FindReferencePoint(point111, point1, point4)
point6 = FindReferencePoint(point222, point2, point5)
point333 = FindReferencePoint(point1, point4, point444)
point666 = FindReferencePoint(point2, point5, point555)
distanceBetweenButtomTwoPoints = Rhino.Distance(point1, point4)
distanceBetweenTopTwoPoints = Rhino.Distance(point2, point5)
var_floorHeight = point2(2) - point1(2)
distanceBetweenPoint1Point2 = Rhino.Distance(point1, point2)
`x, y, z representation of this 8 points
Dim x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4
Dim x5, y5, z5, x6, y6, z6, x7, y7, z7, x8, y8, z8
Dim x9, y9, z9, x10, y10, z10, x11, y11, z11
Dim x12, y12, z12, x13, y13, z13, x14, y14, z14
x1 = point1(0)
y1 = point1(1)
z1 = point1(2)
x2 = point2(0)
y2 = point2(1)
z2 = point2(2)
x3 = point3(0)
y3 = point3(1)
z3 = point3(2)
x4 = point4(0)
y4 = point4(1)
z4 = point4(2)
x5 = point5(0)
y5 = point5(1)
z5 = point5(2)
x6 = point6(0)

```



```

Rhino.AddLayer "layerSpandrelPanel_HKIFC", RGB(190, 190, 190)
Rhino.AddLayer "layerFrittGlass_HKIFC", RGB(142, 154, 154)
Rhino.CurrentLayer("layerProfiles")
'.....
'.....'construction of mullion profiles '.....
'.....
Dim ptForGlassSrf_1(1000) 'collect points for making the glass surface
Dim ptForGlassSrf_2(1000)
Dim i, j, k, h
For i = 0 To var_numberOfHorizontalMullions-1
Rhino.UnselectAllObjects 'make sure no object selected

Dim a1, a2
a1 = var_floorHeight/(var_numberOfHorizontalMullions-1) - 150*(var_numberOfHorizontalMullions-1)/2 + 150/2 'from the equation below

Select Case var_horizontalMullionsStyle
Case "Original_Cesar_Style"
If i = 0 Then var_distanceToLowerPoint = 0
If i = 1 Then var_distanceToLowerPoint = 565
If i = 2 Then var_distanceToLowerPoint = 2550*var_floorHeight/4200
If i = 3 Then var_distanceToLowerPoint = 3150*var_floorHeight/4200
If i = 4 Then var_distanceToLowerPoint = 4199*var_floorHeight/4200
If i = 5 Then var_distanceToLowerPoint = (4199-525)*var_floorHeight/4200
If i = 6 Then var_distanceToLowerPoint = 1165*var_floorHeight/4200
If i = 7 Then var_distanceToLowerPoint = (3150-600)*var_floorHeight/4200
If i > 7 Then Rhino.MessageBox("Not allowed by Cesar_Style!")
Case "Evenly_Distributed_Style"
If i < var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint =
i*distanceBetweenPoint1Point2/(var_numberOfHorizontalMullions-1)
If i = var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint = var_
floorHeight-1
'Height = n*a1+n(n-1)*100/2 ==> Height = na1 + 50n*n - 50n ==> a1 =
Height/n - 50n + 50
Case "Gradient_From_Top_Down_Style"
If i < var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint =
i*a1+i*(i-1)*150/2
If i = var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint = var_
floorHeight-1
Case "Gradient_From_Bottom_Up_Style"
If i > 0 & i < var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint =
var_floorHeight - (i*a1+i*(i-1)*150/2)
If i = 0 Then var_distanceToLowerPoint = 0
If i = var_numberOfHorizontalMullions-1 Then var_distanceToLowerPoint = var_
floorHeight-1
Case Else
Rhino.MessageBox("Out of range!")
End Select 'end of select case var_horizontalMullionsStyle
'.....'first row mullion profile '.....
'.....
'.....'actual function
ptForGlassSrf_1(i) = Mini_typicalHorizontalMullion_1 (Array((x2-x1)*var_dis-
tanceToLowerPoint/var_floorHeight+x1, (y2-y1)*var_distanceToLowerPoint/var_
floorHeight+y1, (z2-z1)*var_distanceToLowerPoint/var_floorHeight+z1), point2,
Array((x6-x3)*var_distanceToLowerPoint/var_floorHeight+x3, (y6-y3)*var_dis-

```

```

tanceToLowerPoint/var_floorHeight+y3, (z6-z3)*var_distanceToLowerPoint/var_
floorHeight+z3), var_HorizontalMullionWidth, var_HorizontalMullionHeight)
objHorizontalMullions_1(i) = Rhino.LastObject
''''''''second row mullion profile
''''''''actual function
ptForGlassSrf_2(i) = Mini_typicalHorizontalMullion_1 (Array((x5-x4)*var_
distanceToLowerPoint/var_floorHeight+x4, (y5-y4)*var_distanceToLowerPoint/
var_floorHeight+y4, (z5-z4)*var_distanceToLowerPoint/var_floorHeight+z4),
point5, Array((x666-x333)*var_distanceToLowerPoint/var_floorHeight+x333, (y666-
y333)*var_distanceToLowerPoint/var_floorHeight+y333, (z666-z333)*var_distanc-
eToLowerPoint/var_floorHeight+z333), var_HorizontalMullionWidth, var_Horizon-
talMullionHeight)
objHorizontalMullions_2(i) = Rhino.LastObject

x7 = (x4 - x1)*var_VerticalMullionWidth/distanceBetweenButtomTwoPoints + x1
y7 = (y4 - y1)*var_VerticalMullionWidth/distanceBetweenButtomTwoPoints + y1
z7 = (z4 - z1)*var_VerticalMullionWidth/distanceBetweenButtomTwoPoints + z1
x8 = (x5 - x2)*var_VerticalMullionWidth/distanceBetweenTopTwoPoints + x2
y8 = (y5 - y2)*var_VerticalMullionWidth/distanceBetweenTopTwoPoints + y2
z8 = (z5 - z2)*var_VerticalMullionWidth/distanceBetweenTopTwoPoints + z2
x9 = (x4 - x1)*(distanceBetweenButtomTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenButtomTwoPoints + x1
y9 = (y4 - y1)*(distanceBetweenButtomTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenButtomTwoPoints + y1
z9 = (z4 - z1)*(distanceBetweenButtomTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenButtomTwoPoints + z1
x10 = (x5 - x2)*(distanceBetweenTopTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenTopTwoPoints + x2
y10 = (y5 - y2)*(distanceBetweenTopTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenTopTwoPoints + y2
z10 = (z5 - z2)*(distanceBetweenTopTwoPoints-var_VerticalMullionWidth)/dis-
tanceBetweenTopTwoPoints + z2

x11 = (x4 - x1)*(50+5)/distanceBetweenButtomTwoPoints + x1 `var_VerticalMul-
lionWidth = 50
y11 = (y4 - y1)*(50+5)/distanceBetweenButtomTwoPoints + y1
z11 = (z4 - z1)*(50+5)/distanceBetweenButtomTwoPoints + z1
x12 = (x5 - x2)*(50+5)/distanceBetweenTopTwoPoints + x2
y12 = (y5 - y2)*(50+5)/distanceBetweenTopTwoPoints + y2
z12 = (z5 - z2)*(50+5)/distanceBetweenTopTwoPoints + z2
x13 = (x4 - x1)*(distanceBetweenButtomTwoPoints-50-5)/distanceBetweenButtomT-
woPoints + x1
y13 = (y4 - y1)*(distanceBetweenButtomTwoPoints-50-5)/distanceBetweenButtomT-
woPoints + y1
z13 = (z4 - z1)*(distanceBetweenButtomTwoPoints-50-5)/distanceBetweenButtomT-
woPoints + z1
x14 = (x5 - x2)*(distanceBetweenTopTwoPoints-50-5)/distanceBetweenTopT-
woPoints + x2
y14 = (y5 - y2)*(distanceBetweenTopTwoPoints-50-5)/distanceBetweenTopT-
woPoints + y2
z14 = (z5 - z2)*(distanceBetweenTopTwoPoints-50-5)/distanceBetweenTopT-
woPoints + z2

point7(0) = x7
point7(1) = y7

```



```

point7(2) = z7
point8(0) = x8
point8(1) = y8
point8(2) = z8
point9(0) = x9
point9(1) = y9
point9(2) = z9
point10(0)=x10
point10(1)=y10
point10(2)=z10
point11(0)=x11
point11(1)=y11
point11(2)=z11
point12(0)=x12
point12(1)=y12
point12(2)=z12
point13(0)=x13
point13(1)=y13
point13(2)=z13
point14(0)=x14
point14(1)=y14
point14(2)=z14
.....
''''''''construction of louvre profiles ''''''''''
.....
''''''''first row louvre profile
If i = 0 Or i = var_numberOfHorizontalMullions-1 Then HKIFC_typicalHorizontalLouvre_1 Array((x2-x1)*var_distanceToLowerPoint/var_floorHeight+x1, (y2-y1)*var_distanceToLowerPoint/var_floorHeight+y1, (z2-z1)*var_distanceToLowerPoint/var_floorHeight+z1), point2, Array((x6-x3)*var_distanceToLowerPoint/var_floorHeight+x3, (y6-y3)*var_distanceToLowerPoint/var_floorHeight+y3, (z6-z3)*var_distanceToLowerPoint/var_floorHeight+z3), var_HorizontalLouvreSize_Width
If i > 0 And i < var_numberOfHorizontalMullions-1 Then HKIFC_typicalHorizontalLouvre_1 Array((x2-x1)*var_distanceToLowerPoint/var_floorHeight+x1, (y2-y1)*var_distanceToLowerPoint/var_floorHeight+y1, (z2-z1)*var_distanceToLowerPoint/var_floorHeight+z1), point2, Array((x6-x3)*var_distanceToLowerPoint/var_floorHeight+x3, (y6-y3)*var_distanceToLowerPoint/var_floorHeight+y3, (z6-z3)*var_distanceToLowerPoint/var_floorHeight+z3), var_HorizontalLouvreSize_Width
objHorizontalLouvres_1(i) = Rhino.LastObject
''''''''second row louvre profile
If i = 0 Or i = var_numberOfHorizontalMullions-1 Then HKIFC_typicalHorizontalLouvre_1 Array((x5-x4)*var_distanceToLowerPoint/var_floorHeight+x4, (y5-y4)*var_distanceToLowerPoint/var_floorHeight+y4, (z5-z4)*var_distanceToLowerPoint/var_floorHeight+z4), point5, Array((x666-x333)*var_distanceToLowerPoint/var_floorHeight+x333, (y666-y333)*var_distanceToLowerPoint/var_floorHeight+y333, (z666-z333)*var_distanceToLowerPoint/var_floorHeight+z333), var_HorizontalLouvreSize_Width
If i > 0 And i < var_numberOfHorizontalMullions-1 Then HKIFC_typicalHorizontalLouvre_1 Array((x5-x4)*var_distanceToLowerPoint/var_floorHeight+x4, (y5-y4)*var_distanceToLowerPoint/var_floorHeight+y4, (z5-z4)*var_distanceToLowerPoint/var_floorHeight+z4), point5, Array((x666-x333)*var_distanceToLowerPoint/var_floorHeight+x333, (y666-y333)*var_distanceToLowerPoint/var_floorHeight+y333, (z666-z333)*var_distanceToLowerPoint/var

```

```

floorHeight+z333), var_HorizontalLouvreSize_Width
objHorizontalLouvres_2(i) = Rhino.LastObject
Next 'end of for loop
'
'Generate vertical mullions profiles'
'
Dim var_VerticalMullionWidthChangingValueNew
var_VerticalMullionWidthChangingValueNew = var_VerticalMullionWidth - 5*verticalMullionWidthTapperRadio
Dim var_VerticalMullionLengthChangingValueNew
var_VerticalMullionLengthChangingValueNew = var_VerticalMullionLength - 6.5*verticalMullionWidthTapperRadio
Rhino.UnselectAllObjects
HKIFC_typicalVerticalMullion_1 point1, point3, point4, var_VerticalMullionWidth, var_VerticalMullionLength
objVerticalMullion_1 = Rhino.LastObject
Rhino.UnselectAllObjects
HKIFC_typicalVerticalMullion_1 point2, point6, point5, var_VerticalMullionWidthChangingValueNew, var_VerticalMullionLengthChangingValueNew
objVerticalMullion_2 = Rhino.LastObject
Rhino.UnselectAllObjects
HKIFC_typicalVerticalMullion_1 point4, point333, point1, var_VerticalMullionWidth, var_VerticalMullionLength
objVerticalMullion_3 = Rhino.LastObject
Rhino.UnselectAllObjects
HKIFC_typicalVerticalMullion_1 point5, point666, point2, var_VerticalMullionWidthChangingValueNew, var_VerticalMullionLengthChangingValueNew
objVerticalMullion_4 = Rhino.LastObject
'
'Construct generative component'
'
Rhino.CurrentLayer("layerHorizontalMullion_HKIFC")
'Horizontal Mullions
Rhino.UnselectAllObjects
For i = 0 To var_numberOfHorizontalMullions-1
LoftSurfaceByNumberOfProfile 2, objHorizontalMullions_1(i), objHorizontalMullions_2(i), Null, Null
Next 'end of for loop
objHorizontalMullions_4(i), objHorizontalMullions_2(i)
Rhino.CurrentLayer("layerHorizontalLouvre_HKIFC")
'Horizontal Louvres
Rhino.UnselectAllObjects
For i = 0 To var_numberOfHorizontalMullions-1
LoftSurfaceByNumberOfProfile 2, objHorizontalLouvres_1(i), objHorizontalLouvres_2(i), Null, Null
Next 'end of for loop

Rhino.CurrentLayer("layerVerticalMullion_HKIFC")
'Vertical Mullions
LoftSurfaceByNumberOfProfile 2, objVerticalMullion_1, objVerticalMullion_2, Null, Null
LoftSurfaceByNumberOfProfile 2, objVerticalMullion_3, objVerticalMullion_4, Null, Null
'
'make the glass

```

```

Rhino.CurrentLayer("layerSpandrelPanel_HKIFC")
''''''''Spandrel Panel
Dim arrPtForSrfPt_1(3)
arrPtForSrfPt_1(0)= Rhino.PointCoordinates(ptForGlassSrf_1(0))
arrPtForSrfPt_1(1)= Rhino.PointCoordinates(ptForGlassSrf_1(1))
arrPtForSrfPt_1(2)= Rhino.PointCoordinates(ptForGlassSrf_2(1))
arrPtForSrfPt_1(3)= Rhino.PointCoordinates(ptForGlassSrf_2(0))
Rhino.AddSrfPt arrPtForSrfPt_1
Rhino.CurrentLayer("layerTransparentGlass_HKIFC")
''''''''Transparent Glass
Dim arrPtForSrfPt_2(3)
arrPtForSrfPt_2(0)= Rhino.PointCoordinates(ptForGlassSrf_1(1))
arrPtForSrfPt_2(1)= Rhino.PointCoordinates(ptForGlassSrf_1(2))
arrPtForSrfPt_2(2)= Rhino.PointCoordinates(ptForGlassSrf_2(2))
arrPtForSrfPt_2(3)= Rhino.PointCoordinates(ptForGlassSrf_2(1))
Rhino.AddSrfPt arrPtForSrfPt_2
Rhino.CurrentLayer("layerSpandrelPanel_HKIFC")
''''''''Spandrel Panel
Dim arrPtForSrfPt_3(3)
arrPtForSrfPt_3(0)= Rhino.PointCoordinates(ptForGlassSrf_1(2))
arrPtForSrfPt_3(1)= Rhino.PointCoordinates(ptForGlassSrf_1(3))
arrPtForSrfPt_3(2)= Rhino.PointCoordinates(ptForGlassSrf_2(3))
arrPtForSrfPt_3(3)= Rhino.PointCoordinates(ptForGlassSrf_2(2))
Rhino.AddSrfPt arrPtForSrfPt_3
''''''''Spandrel Panel
Dim arrPtForSrfPt_4(3)
arrPtForSrfPt_4(0)= Rhino.PointCoordinates(ptForGlassSrf_1(3))
arrPtForSrfPt_4(1)= Rhino.PointCoordinates(ptForGlassSrf_1(4))
arrPtForSrfPt_4(2)= Rhino.PointCoordinates(ptForGlassSrf_2(4))
arrPtForSrfPt_4(3)= Rhino.PointCoordinates(ptForGlassSrf_2(3))
Rhino.AddSrfPt arrPtForSrfPt_4
Rhino.CurrentLayer("layerProfiles")
Rhino.UnselectAllObjects
End Sub 'end of HKIFC_component_1

Function HKIFC_typicalHorizontalMullion_1 (point1, point2, point3, horizontal-
MullionWidth, horizontalMullionHeight)

Dim var_MullionWidth, var_MullionHeight
var_MullionWidth = horizontalMullionWidth - 100
var_MullionHeight = horizontalMullionHeight - 237
Dim point, x, y, z
point = Array(0,0,0)
x = point(0)
y = point(1)
z = point(2)

Dim pt1, pt2, pt3, pt4, pt5, pt6, pt7, pt8, pt9, pt10, _
pt11, pt12, pt13, pt14, pt15, pt16, pt17, pt18, pt19, ptForGlass
pt1 = Array(x+15.256,y+35.5522+var_MullionHeight,z)
pt2 = Array(x+25,y+35.5522+var_MullionHeight,z)
pt3 = Array(x+50+var_MullionWidth/2/2,y+35.5522+var_MullionHeight,z)
pt4 = Array(x+75+var_MullionWidth/2,y+35.5522+var_MullionHeight,z)

```

```

pt5 = Array(x+84.744+var_MullionWidth/2,y+35.5522+var_MullionHeight,z)
pt6 = Array(x+96.1748+var_MullionWidth/2,y-34.0334+var_MullionHeight,z)
pt7 = Array(x+100+var_MullionWidth/2,y-104.448+var_MullionHeight,z)
pt8 = Array(x+100+var_MullionWidth/2,y-114.448,z)
pt9 = Array(x+76+var_MullionWidth/2,y-114.448,z)
pt10 = Array(x+76+var_MullionWidth/2,y-151.451,z)
pt11 = Array(x+100+var_MullionWidth/2,y-151.451,z)
pt12 = Array(x+100+var_MullionWidth/2,y-176.451,z)
pt13 = Array(x,y-176.451,z)
pt14 = Array(x,y-151.451,z)
pt15 = Array(x+24,y-151.451,z)
pt16 = Array(x+24,y-114.448,z)
pt17 = Array(x,y-114.448,z)
pt18 = Array(x,y-104.448+var_MullionHeight,z)
pt19 = Array(x+3.82525,y-34.0334+var_MullionHeight,z)

ptForGlass = Array(x+var_MullionWidth/2,y-132.950+var_MullionHeight,z)

Dim strPt1, strPt2, strPt3, strPt4, strPt5, strPt6, strPt7, strPt8, strPt9, strPt10, _
    strPt11, strPt12, strPt13, strPt14, strPt15, strPt16, strPt17, strPt18, strPt19
strPt1 = Rhino.Pt2Str(pt1)
strPt2 = Rhino.Pt2Str(pt2)
strPt3 = Rhino.Pt2Str(pt3)
strPt4 = Rhino.Pt2Str(pt4)
strPt5 = Rhino.Pt2Str(pt5)
strPt6 = Rhino.Pt2Str(pt6)
strPt7 = Rhino.Pt2Str(pt7)
strPt8 = Rhino.Pt2Str(pt8)
strPt9 = Rhino.Pt2Str(pt9)
strPt10 = Rhino.Pt2Str(pt10)
strPt11 = Rhino.Pt2Str(pt11)
strPt12 = Rhino.Pt2Str(pt12)
strPt13 = Rhino.Pt2Str(pt13)
strPt14 = Rhino.Pt2Str(pt14)
strPt15 = Rhino.Pt2Str(pt15)
strPt16 = Rhino.Pt2Str(pt16)
strPt17 = Rhino.Pt2Str(pt17)
strPt18 = Rhino.Pt2Str(pt18)
strPt19 = Rhino.Pt2Str(pt19)

Dim strPtForGlass
strPtForGlass = Rhino.Pt2Str(ptForGlass)

Dim arrPlaneNew(2)
arrPlaneNew(0) = Array(point1(0),point1(1),point1(2))
arrPlaneNew(1) = Array(point2(0),point2(1),point2(2))
arrPlaneNew(2) = Array(point3(0),point3(1),point3(2))
aaastrView = Rhino.CurrentView
Rhino.ViewCPlane aaastrView, arrPlaneNew
Rhino.Command("_point " & strPtForGlass & " ")
HKIFC_typicalHorizontalMullion_1 = Rhino.LastObject

Dim line1, line2, line3, line4, line5, line6, line7, line8, line9, line10
Dim line11, line12, line13, line14, line15, line16
Rhino.Command("_line " & strPt1 & " " & strPt5 & " ")

```

```

line1 = Rhino.LastObject
Rhino.Command("_InterpCrv " & strPt5 & " " & strPt6 & " " & strPt7 & " " & "Enter ")
line2 = Rhino.LastObject
Rhino.Command("_line " & strPt7 & " " & strPt8 & " ")
line3 = Rhino.LastObject
Rhino.Command("_line " & strPt8 & " " & strPt9 & " ")
line4 = Rhino.LastObject
Rhino.Command("_line " & strPt9 & " " & strPt10 & " ")
line5 = Rhino.LastObject
Rhino.Command("_line " & strPt10 & " " & strPt11 & " ")
line6 = Rhino.LastObject
Rhino.Command("_line " & strPt11 & " " & strPt12 & " ")
line7 = Rhino.LastObject
Rhino.Command("_line " & strPt12 & " " & strPt13 & " ")
line8 = Rhino.LastObject
Rhino.Command("_line " & strPt13 & " " & strPt14 & " ")
line9 = Rhino.LastObject
Rhino.Command("_line " & strPt14 & " " & strPt15 & " ")
line10 = Rhino.LastObject
Rhino.Command("_line " & strPt15 & " " & strPt16 & " ")
line11 = Rhino.LastObject
Rhino.Command("_line " & strPt16 & " " & strPt17 & " ")
line12 = Rhino.LastObject
Rhino.Command("_line " & strPt17 & " " & strPt18 & " ")
line13 = Rhino.LastObject
Rhino.Command("_InterpCrv " & strPt18 & " " & strPt19 & " " & strPt1 & " " & "Enter ")
line14 = Rhino.LastObject

```

```

Dim cProfile(13)
cProfile(0)=line1
cProfile(1)=line2
cProfile(2)=line3
cProfile(3)=line4
cProfile(4)=line5
cProfile(5)=line6
cProfile(6)=line7
cProfile(7)=line8
cProfile(8)=line9
cProfile(9)=line10
cProfile(10)=line11
cProfile(11)=line12
cProfile(12)=line13
cProfile(13)=line14
Rhino.SelectObjects cProfile
Rhino.Command "_Join "
Dim jointedVerticalMullion
End Function 'end of HKIFC_typicalHorizontalMullion_1

```

```

Function FindDivisionPoints(curvel, numofDivisions)
Dim numofPoint, numofPoints
Dim i, convertPoint
ReDim convertedPoints(100000)

```

```

numOfPoints = Rhino.DivideCurve (curve1, numOfDivisions)
For Each numofPoint In numOfPoints
    Rhino.AddPoint numofPoint
Next
For i=0 To numOfDivisions
    Rhino.Command ("_cplane " & "World" & " " & "Top" & " ")
    Rhino.Command ("_point " & Pt2Str(numofPoints(i)) & " ")
    convertedPoint = Rhino.LastObject
    convertedPoints(i)=Rhino.PointCoordinates(convertedPoint)
Next
FindDivisionPoints = convertedPoints
End Function 'end of FindDivisionPoints

```

```

Function FindReferencePoint(point1, point2, point3)
Dim lineAngle13, convectOrConcave, checkAligned, distPt2Pt3
Dim distPt1Pt2, midPt1Pt3, distPt2MidPt1Pt3
Dim convertPoint, convertedPoints
distPt1Pt2 = Rhino.Distance(point1, point2)
distPt2Pt3 = Rhino.Distance(point2, point3)
midPt1Pt3 = Array((point3(0)-point1(0))*distPt1Pt2/
(distPt1Pt2+distPt2Pt3)+point1(0), (point3(1)-point1(1))*distPt1Pt2/
(distPt1Pt2+distPt2Pt3)+point1(1), (point3(2)-point1(2))*distPt1Pt2/
(distPt1Pt2+distPt2Pt3)+point1(2)) 'not mid point, but propotional
distPt2MidPt1Pt3 = Rhino.Distance(point2, midPt1Pt3)
checkAligned = CheckPointsAligned (point1, point2, point3)
If checkAligned = True Then 'check if aligned
    lineAngle13 = Rhino.Angle(point1, point3)
    convertPoint = Array((point2(0)-(1000000000+distPt2Pt3)*Sin(lineAngle13(0)*
0.0174532925)), (point2(1)+(1000000000+distPt2Pt3)*Cos(lineAngle13(0)*0.01745
32925)), point2(2))
Else 'if not aligned then do
something
    convectOrConcave = CheckConcaveOrConvect (point1, point2, point3)
    If convectOrConcave = True Then 'convect
        convertPoint = Array(((midPt1Pt3(0)-point2(0))*(1000000000+distPt2MidPt1P
t3)/distPt2MidPt1Pt3+point2(0)), ((midPt1Pt3(1)-point2(1))*(1000000000+distPt
2MidPt1Pt3)/distPt2MidPt1Pt3+point2(1)), ((midPt1Pt3(2)-point2(2))*(100000000
0+distPt2MidPt1Pt3)/distPt2MidPt1Pt3+point2(2))) 'extention of mid point of
point1 and point3
    Else convertPoint = Array(2*point2(0)-(point1(0)+point3(0))/2,
2*point2(1)-(point1(1)+point3(1))/2, 2*point2(2)-(point1(2)+point3(2))/2)
'mid point of point1 and point3
    End If
End If
Rhino.Command ("_cplane " & "World" & " " & "Top" & " ")
Rhino.Command ("_point " & Pt2Str(convertPoint) & " ")
convertedPoints = Rhino.LastObject
FindReferencePoint = Rhino.PointCoordinates(convertedPoints)
End Function 'end of FindReferencePoint

```

```

Function FindExtraPointFromEdge(point1, point2)
Dim extraPointFromEdge
Dim convertPoint
Dim convertedPoints

```



```

ReDim findDiagridPoints1(2*numOfDivisions+1)
Dim i
Dim convertPoint
ReDim convertedPoints(1000)
numOfPoints = Rhino.DivideCurve (curve1, numOfDivisions)
For Each numOfPoint In numOfPoints
    Rhino.AddPoint numOfPoint
Next

For i=0 To numOfDivisions
Rhino.Command ("_cplane " & "World" & " " & "Top" & " ")
Rhino.Command ("_point " & Pt2Str(numOfPoints(i)) & " ")
convertedPoint = Rhino.LastObject
convertedPoints(i)=Rhino.PointCoordinates(convertedPoint)
Next
findDiagridPoints1(0) = convertedPoints(0)

For i=0 To numOfDivisions-1
pt1 = convertedPoints(i)
pt2 = convertedPoints(i+1)
distPt1Pt2 = Rhino.Distance (pt1, pt2)
If i=0 Then
divPoint1 = Array((pt2(0)-pt1(0))*2*distance1/distPt1Pt2+pt1(0), (pt2(1)-
pt1(1))*2*distance1/distPt1Pt2+pt1(1), (pt2(2)-pt1(2))*2*distance1/
distPt1Pt2+pt1(2))
divPoint2 = Array((pt2(0)-pt1(0))*(distPt1Pt2-distance1)/distPt1Pt2+pt1(0),
(pt2(1)-pt1(1))*(distPt1Pt2-distance1)/distPt1Pt2+pt1(1), (pt2(2)-
pt1(2))*(distPt1Pt2-distance1)/distPt1Pt2+pt1(2))
End If
If i>0 And i<numOfDivisions-1 Then
divPoint1 = Array((pt2(0)-pt1(0))*distance1/distPt1Pt2+pt1(0), (pt2(1)-
pt1(1))*distance1/distPt1Pt2+pt1(1), (pt2(2)-pt1(2))*distance1/
distPt1Pt2+pt1(2))
divPoint2 = Array((pt2(0)-pt1(0))*(distPt1Pt2-distance1)/distPt1Pt2+pt1
(0), (pt2(1)-pt1(1))*(distPt1Pt2-distance1)/distPt1Pt2+pt1(1), (pt2(2)-
pt1(2))*(distPt1Pt2-distance1)/distPt1Pt2+pt1(2))
End If
If i=numOfDivisions-1 Then
divPoint1 = Array((pt2(0)-pt1(0))*distance1/distPt1Pt2+pt1(0), (pt2(1)-
pt1(1))*distance1/distPt1Pt2+pt1(1), (pt2(2)-pt1(2))*distance1/
distPt1Pt2+pt1(2))
divPoint2 = Array((pt2(0)-pt1(0))*(distPt1Pt2-2*distance1)/distPt1Pt2+pt1(0),
(pt2(1)-pt1(1))*(distPt1Pt2-2*distance1)/distPt1Pt2+pt1(1), (pt2(2)-
pt1(2))*(distPt1Pt2-2*distance1)/distPt1Pt2+pt1(2))
End If
Rhino.AddPoint divPoint1
Rhino.AddPoint divPoint2
findDiagridPoints1(2*i+1) = divPoint1
findDiagridPoints1(2*i+2) = divPoint2
Next
findDiagridPoints1(2*numOfDivisions+1) = convertedPoints(numOfDivisions) 'add
last point

FindDiagridPoints = findDiagridPoints1
End Function 'end of FindDiagridPoints

```


Appendix C: Table of Figures

Figure 1: Dundas Entertainment Center

Figure 2: Informational & Mediatized Architectural Surfaces

Figure 3: The Canadian Opera House

Figure 4: The Tulsa Arena

Figure 5: Dundas Entertainment Center

Figure 6: Reflected Panels in the Auditorium of the Canadian Opera House

Figure 7: Balcony Front in the Auditorium of the Canadian Opera House

Figure 8: The Tulsa Arena

Figure 9: The Tulsa Arena

Figure 10: DG Bank by Frank Gehry

- image taken from Branko Kolarevic's Digital Production

Figure 11: British Museum Great Court by Norman Foster

- image taken from Branko Kolarevic's Digital Production

Figure 12: Triangulation of a doubly-curved surface

- image taken from Branko Kolarevic's Digital Production

Figure 13: The Sketchpad System by Ivan Sutherland

- image taken from Yehuda E. Kalay's Architecture's New Media

Figure 14: URBAN5

- image taken from Yehuda E. Kalay's Architecture's New Media

Figure 15: AutoCAD by AutoDesk

Figure 16: Guggenheim Museum by Frank Gehry

- image taken from www.greatbuildings.com/buildings/Guggenheim_Bilbao.html

Figure 17: Digital Project by The Gehry Technology

- image taken from www.gehrytechnologies.com

Figure 18: Swiss Re by Norman Foster

- image taken from Guy Nordenson's Tall Buildings

Figure 19: New York Times Headquarters by Frank Gehry/SOM

- image taken from Guy Nordenson's Tall Buildings

Figure 20: KAAD by G.Crrara et al.

- *image taken from Yehuda E. Kalay's Architecture's New Media*

Figure 21: The Twisted Skyscraper

Figure 22: The Twisted Skyscraper

Figure 23: HKIFC by Cesar Pelli

Figure 24: HKIFC by Cesar Pelli

Figure 25: HKIFC by Cesar Pelli

Figure 26: The Petronas Tower by Cesar Pelli

Figure 27: The Petronas Tower by Cesar Pelli

Figure 28: The Petronas Tower by Cesar Pelli

Figure 29: The Cira Tower by Cesar Pelli

Figure 30: The Cira Tower by Cesar Pelli

Figure 31: The Hearst Corporation

Figure 32: The Hearst Corporation

Figure 33: The Hearst Corporation

Figure 34: Tree Structure of the Program

Figure 35: Sub-Tree Structure of the Program

Figure 36: Profiles Finding Method

Figure 37: Typical Plan Profile

Figure 38: Transformational Rules

Figure 39: Massing Construction Procedure

Figure 40: User Interface for User interaction with the program

Figure 41: 3D Printed Models

Figure 42: 3D Models of Original Design

Figure 43: Original Design Projects

Figure 44: Design Variations of the Petronas Generated by the Proposed System

Figure 46: Design Variations of the Petronas Tower Generated by the Proposed System

Figure 48: 3D Printed Models Showing Variations of the Petronas Tower

Figure 49: 3D Printed Models Showing Design Variations of the Petronas Tower

Figure 50: Design Variations of HKIFC Generated by the Proposed System

Figure 52: Design Variations of Generated by the Proposed System

Figure 54: 3D Printed Models Showing Design Variations of HKIFC

Figure 55: 3D Printed Models Showing Design Variations of HKIFC

Figure 56: Design Variations of the Cira Tower Generated by the Proposed System

Figure 58: Design Variations of the Cira Tower Generated by the Proposed System

Figure 60: 3D Printed Models Showing Design Variations of the Cira Tower

Figure 61: 3D Printed Models Showing Design Variations of the Cira Tower

Figure 62: Design Variations of the Hearst Corporation Generated by the Proposed System

Figure 64: Design Variations of the Hearst Corporation Generated by the Proposed System

Figure 66: 3D Printed Models Showing Design Variations of the Hearst Corporation

Figure 67: 3D Printed Models Showing Design Variations of the Hearst Corporation

Figure 68: Constructing Massing in a Site Context

Figure 69: Constructing Variations on a Same Site

Figure 71: 3D Printed Models Showing Forrest of Towers of the Variations Shown
at Previous pages

Figure 72: Types of Details

Figure 73: Diagrid Constructed on a Surface

Figure 74: Constructing Details on Selected Levels

Figure 75: Construct Detail in Different Resolution Settings

Figure 76: Incorporation of Massing and Details

Figure 77: 3D Printed Models Showing a New Design: the Drum-Shaped Tower

Figure 78: 3D Printed Models Showing a New Design: the Drum-Shaped Tower

Figure 79: 3D Printed Models Showing the Details on the Top Levels of the
Drum-Shaped Tower

Figure 80: 3D Printed Models Showing the Details on the lower Levels of the
Drum-Shaped Tower

Figure 81: 3D Printed Models Showing a New Design: the New Twisted Skyscraper

Figure 82: 3D Printed Models Showing a New Design: the New Twisted Skyscraper

Figure 83: 3D Printed Models Showing the Details on the Top Levels of the
New Twisted Skyscraper

Figure 84: 3D Printed Models Showing the Details on the Lower Levels of the
New Twisted Skyscraper

Bibliography:

- Ali, Mir M. *Art of the Skyscraper—The Genius of Fazlur Khan*. Rizzoli International Publications, Inc. New York. 2001
- Alofsin, Anthony. *Prairie Skyscraper—Frank Lloyd Wright's Price Tower*. Rizzoli International Publications, Inc. New York. 2005
- Borg, Jonathan C., Farrugia, Phillip J. and Camilleri, Kenneth P. *Knowledge Intensive Design Technology*. Klumwer Academic Publishers. Norwell, Massachusetts. 2004.
- Conway, Donald J. *Human Response to Tall Buildings*. Dowden, Hutchinson & Ross, Inc. Stroudsburg, Pennsylvania. 1997.
- Crosbie, Michael J. *Curtain Walls*. Birkhauser-Publishers for Architecture. Basel, Switzerland. 2006.
- Eisele, Johann & Kloft, Ellen. *High-Rise Manual*. Birkhauser-Publishers for Architecture. Basel, Switzerland. 2003.
- Jone, Robin & Maynard, Clive & Stewart, Ian. *The Art of Lisp Programming*. Springer-Verlag, London, Britian. 1990.
- Kalay, Yehuda E. *Architectures New Media*. The MIT Press, Cambridge, Massachusetts. 2005.
- Leach, Neil. *Design for a Digital World*. Wiley-Academy. London, UK. 2002.
- Moudry, Roberta. *The American Skyscraper—Cultural Histories*. Cambridge University Press. New York, NY. 2005
- Negroponte, Nicholas. *Reflections on Computer Aids to Design and Architecture*. Mason/Charter Publishers, Inc. London, England. 1975.
- Nordenson, Guy. *Tall Buildings*. Museum of Modern Art. New York. 2003.
- Ockman, Joan. *Tower and Office—From Modernist Theory to Contemporary Practice*. The MIT Press. Cambridge, Massachusetts. 2003.
- Pena, William M. and Parshall, Steven A. *Problem Seeking—An Architectural Programming Primer*. John Wiley & Sons, Inc. New York, NY. 2001.
- Peng, Chengzhi. *Design through Digital Interaction*. Intellect Books. Bristol, UK. 2001.

- Perex-Gomez, Alberto & Pelletier Louise. *Architectural Representation and the Perspective Hinge*. The MIT Press, Cambridge, Massachusetts. 1997.
- Phiri, Michael. *Information Technology in Construction Design*. Thomas Telford Publishing. London, UK. 1999.
- Schmitt, Gerhard. *Information Architecture*. Birkhauser Publishers for Architecture. Basel, Switzerland. 1999.
- Serraino, Pierluigi. *History of FormZ*. Birkhauser Publishers for Architecture. Basel, Switzerland. 2002.
- Standiford, Kevin. *AutoLisp to VisualLisp Design Solutions for AutoCad*. Autodesk Press. Albany, NY. 2001.
- Stefanuk, Vadim and Kaijiri, Kenji. *Knowledge-Based Software Engineering*. IOS Press. Amsterdam, The Netherlands. 2004.
- Szalapaj, Peter. *Contemporary Architecture and the Digital Design Process*. Elsevier Architectural Press. Burlington, Massachusetts. 2005.
- Vries, Bauke de & Leeuwen, Jos Van & Achten, Henri. *Computer Aided Architectural Design Futures 2001*. Kluwer Academic Publishers. Dordrecht, The Netherlands. 2001.
- Wright, Gordon. "Building on Tradition" in *Building Design and Construction*. 2005.
URL: <http://www.bdcnetwork.com/article/CA6161495.html>.