A PROTOTYPE SKETCH-BASED ARCHITECTURE MODELING SYSTEM WITH TRADITIONAL DESIGN HABITS

by

Jun Yu
Bachelor of Engineering, University of Electronic Science and Technology of China, 2002
Bachelor of Engineering, Chongqing University, 1993

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School of Computing Science

© Jun Yu 2008
SIMON FRASER UNIVERSITY
Summer 2008

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
APPROVAL

Name: Jun Yu

Degree: Master of Science

Title of thesis: A PROTOTYPE SKETCH-BASED ARCHITECTURE MODELING SYSTEM WITH TRADITIONAL DESIGN HABITS

Examining Committee: Dr. Lou Hafer
Chair

Dr. Richard Zhang
Assistant Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Arthur (Ted) Kirkpatrick
Assistant Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Rob Woodbury
Professor, Interactive Arts + Technology
Simon Fraser University
SFU Examiner

Date Approved: June 13, 2008
Declaration of Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection (currently available to the public at the "Institutional Repository" link of the SFU Library website at: <http://ir.lib.sfu.ca>) and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

While licensing SFU to permit the above uses, the author retains copyright in the thesis, project or extended essays, including the right to change the work for subsequent purposes, including editing and publishing the work in whole or in part, and licensing other parties, as the author may desire.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
Abstract

Widely used for architectural design, CAD systems such as AutoCAD, 3ds MAX, provide powerful modeling functionality. However, they are often too complex for an architect to use, especially during the conceptualization stage involving primarily sketching of design ideas. Also, interactions with conventional CAD systems are typically driven by mouse and keyboard, but these devices cannot be used as freely as a pencil. For an architect, a tablet PC with stylus provides an ideal interface for design sketches. In this report, we study sketch-based architectural design and modeling and develop a prototype system which respects simplicity and traditional conceptual design habits. Our system provides a seamlessly integrated design/draw environment with intuitive and focused interaction built in, both in an attempt to provide a design environment that closely conforms to the architects' conceptual design habits.

Keywords: Sketch; Modeling; Prototype; Design
Acknowledgments

I would like to express my gratitude to Dr. Richard Zhang who has been more than a supervisor throughout my studies. He has always provided me with guidance, motivation, and support. Having the chance to work with him was an honour. I would also like to take this opportunity to thank the fellow students at the GrUVi lab for lending their helping hands whenever I desperately needed help. I give special acknowledgement to my parents who were of great influence to me throughout my life. I thank them for their unconditional love and support. Last but not least, I owe special thanks to my wife Cindy for her continuous support and encouragement. Without them, I would not have been able to achieve my goal.
Contents

Approval ii
Abstract iii
Acknowledgments iv
Contents v
List of Tables viii
List of Figures ix
List of Programs xvi

1 Introduction 1
  1.1 Motivations and Applications 2
  1.2 System Features 4
  1.3 Thesis Organization 7

2 Related Works 8
  2.1 Computer-aided design (CAD) modeling software used in architectural design industry 8
  2.2 Academic research on sketch-based modeling 9
    2.2.1 General topics on sketch-based modelling 9
    2.2.2 Using sketch to create free-form objects 11
    2.2.3 Create featured objects based on sketch 12
    2.2.4 Reconstruct objects using sketch 13
2.2.5 Modify objects using sketches as guidance ........................................ 14
2.2.6 Sketch-based architectural modelling ............................................. 15

3 Design Habits ..................................................................................... 17
  3.1 Descriptive geometry ..................................................................... 18
  3.2 Design work flow ......................................................................... 19

4 Overview of System ........................................................................... 22
  4.1 Supported model creation .............................................................. 22
  4.2 Hardware Interface ..................................................................... 24
  4.3 Mouse Operations ....................................................................... 24
  4.4 Software Interface ....................................................................... 25
  4.5 Basic Modeling Functions ............................................................ 27

5 Extrusion Method ............................................................................. 29
  5.1 Creation operation in Extrusion module ........................................ 29
      5.1.1 Sketch capture, recording and displaying .................................. 29
      5.1.2 Line segment recognition ....................................................... 30
      5.1.3 Cubic Bezier curve construction from sketches ...................... 31
      5.1.4 Poly-line construction .......................................................... 34
      5.1.5 Polygon triangulation and extrusion ....................................... 34
  5.2 Modification operation in Extrusion module .................................. 36
      5.2.1 Colors used on the Extrusion elements .................................. 36
      5.2.2 Extrusion element Selection/Deselection .................................. 36
      5.2.3 Extrusion element and control point Movement in 2D workspace 37
      5.2.4 Extrusion element Connection ............................................... 38
      5.2.5 Extrusion surface and object Movement in 3D workspace .......... 39
  5.3 Modeling aspects in Extrusion module ......................................... 40
      5.3.1 Sketch-retained design process .............................................. 40
      5.3.2 Cubic Bezier curve modification .............................................. 41
      5.3.3 Curve smoothness ................................................................. 41
      5.3.4 Flexible feature tolerance at curve joints .................................. 42
      5.3.5 2D/3D dynamic synchronization ............................................. 42
6 Primitive Method

6.1 Primitive model classification ........................................... 45
6.2 Common properties about Primitive models .......................... 46
   6.2.1 Control frame used on Primitive models ......................... 46
   6.2.2 Colors and line types used on control frames .................. 47
   6.2.3 Control point auto snap features ............................... 48
   6.2.4 Model precision adjustment ...................................... 49
6.3 Creation operation in Primitive module ............................... 49
6.4 Modification operation in Primitive module ........................... 51
   6.4.1 Primitive object selection ..................................... 51
   6.4.2 Column modification ........................................... 52
   6.4.3 Ellipsoid modification ......................................... 54
6.5 Modeling aspects in Primitive module ................................. 56
   6.5.1 Primitive object 2D center modification ......................... 56
   6.5.2 Column object horizontal surface selection auto-switch .... 56
   6.5.3 See-through and Footprint observation features for Ellipsoid objects .......................... 57
6.6 An example of modifying Ellipsoid objects based on half-axis observation .......................... 58

7 Design Examples ................................................................. 60

8 Conclusion and Future Work .................................................. 66

Bibliography ........................................................................... 70
List of Tables

4.1 **Function key mapping between a mouse and a Tablet PC stylus.** Only these simple stylus functions listed in this table are needed to fulfill the whole operation in our current system. This shows the high efficiency of using the stylus. On the other hand, we can notice that only a very limited function combination of the Tablet PC stylus is used in our current system. That means the function of the stylus has a great potential to be extended when new operation features are added. ........................................ 25

4.2 **Mouse Operations used in our modeling system.** Different mouse operations in different program modes correspond to different functions. . . . 26
List of Figures

1.1 A model of Master architect Frank Lloyd Wright’s “Fallingwater Villa” created by our prototype system. (a) Sketching in 2D. (b) Assembly and extrusion. (c) Modification (colored polygons). (d) Final product. The figure shows the work-flow of our system and illustrates the system functionality of sketching, assembly, and modification. 3

3.1 Architectural drawings. (a) A simple building model. (b) Top view of the building. (c) Site map. 21

3.2 Sketches for designing the building. (a) Sketch-1. (b) Sketch-2. (c) Sketch-3. 21

3.3 Work flow of designing a building. (a) Step-1. (b) Step-2. (c) Step-3. (d) Step-4. 21

4.1 Models supported by our system. Users can create and modify Extrusion and Primitive models in this prototype system. 23

4.2 Software interface of our sketch-based modeling system. Extrusion and Primitive models can be created in this program layout. Users can switch 2D/3D modes upon their requests. Pan, Zoom in/out and Rotation operations are supported. 27

5.1 A stroke is constructed by a continuous sketch. Strokes are used to recognize our sketches to line segments or curves. Red points on the sketch illustrate captured screen points. 30
5.2  (a) **Line segment recognition.** Blue rectangles illustrate the test envelopes given by a user-defined threshold (thickness of the envelope). The threshold can be automatically adjusted based on stroke length. If all captured points along the stroke fall inside the corresponding envelope, a line segment is recognized.  
(b) **Recognition of axis-aligned line segments.** When we want to draw an axis-aligned line segment, it is hard to keep the stroke drawn absolutely horizontal or vertical. With acceptable tolerance, a line close to being axis-aligned will be converted to one automatically. 

5.3  (a) **Single piece cubic Bezier curve.** Control points $p_1$, $p_2$, $p_3$ and $p_4$ determine this Bezier curve. In order to manipulate this curve, we should find all these four control points.  
(b) **Single piece cubic Bezier curve subdivision.** We use a binary tree data structure to store control points. Data on the top of leaves are what we need to draw cubic Bezier curves. Data on the root are control points used to manipulate cubic Bezier curves. 

5.4  **A new line segment finds one or two end intersections.** (a) A line segment intersects with another line segment. (b) A line segment intersects with a poly-line. (c) A line segment intersects with other two line segments. (d) A line segment intersects with a line segments and a poly-line. (e) A line segment intersects with two poly-lines. (f) A line segment intersects with a poly-line and close it. 

5.5  **2D polygon triangulation.** (a) maintains the Delaunay triangulation criterion while (b) does not. (c) and (d) are two triangulated polygons computed by our system. 

5.6  **Extruding 2D polygons to 3D objects.** (a) and (b) show 2 models created by the Extrusion operation. 

5.7  **Different colors used on different geometric elements.** (a) **Line Segment.** (b) **Open Poly-line.** (c) **Closed Polygon.** (d) Unselected **3D Extruded Entity.** (e) Surface selected **3D Extruded Entity.** (f) Body selected **3D Extruded Entity.** 

5.8  **Extrusion element selection in 2D workspace.** (a) **Line Segment.** (b) **Open Poly-line.** (c) **Closed Polygon or 3D extrusion entity.**
5.9 **Extrusion** element and control point **Movement** in 2D workspace. (a)-(c) Move control points of *Line Segment*, *Open Poly-line* and *Closed Polygon* or 3D extrusion entity. The moved control points are showed as solid red squares. (d)-(f) Move *Line Segment*, *Open Poly-line* and *Closed Polygon* or 3D extrusion entity. .......................... 37

5.10 **Extrusion** element **Connection**. (a) A line segment intersects with a line segment. (b) A line segment intersects with a poly-line. (c) A line segment intersects with two line segments. (d) A line segment intersects with a line segments and a poly-line. (e) A poly-line intersects with another poly-line. (f) Self-close a poly-line. ................. 38

5.11 **Extrusion** surface and object **Movement** in 3D workspace. (a) Move the top surface to change the height/thickness of an *Extrusion* object. (b) Move an entire *Extrusion* object in the space to change its elevation. ......... 39

5.12 **Sketch-retained design process**. The *Fallingwater Villa* model (a) is displayed in translucent mode together with the sketches on the floor plan. An imaginary architectural model and the design sketches showed in (b). ......... 40

5.13 **Cubic Bezier curve modification**. In this figure, two examples showed in (a) and (b) illustrate the cubic Bezier curve modification operation. .............. 41

5.14 **Curve smoothness**. This figure shows an extrusion object with a piece of curved side surface created using 3 smoothness options corresponding to cubic Bezier curve subdivision level 3 (8 line segments), 4 (16 line segments) and 5 (32 line segments) in (a), (b) and (c) respectively. .............. 42

5.15 **Flexible feature tolerance at curve joints**. The sharp feature is kept at the joint of two Bezier curves in (a), and (b) depicts how to make the joint keep the $C^1$ or $G^1$ continuity. .................. 43

5.16 **Flexible feature tolerance at curve joints**. An initial extrusion object is showed in (a), while the modified 2D and 3D views showed in (b) after changing the base polygon shape in 2D workspace. .............. 43

6.2 **Ellipsoid-like primitive objects.** (a)-(f) Objects created using different half-axes. All models are of the moderate smoothness precision. The six half-axes of an ellipsoid-like object can be of different lengths, but they are required to converge to a same point, the center of the ellipsoid-like object. All these objects conform to same requirements, although they may look quite different from each other.

6.3 **Primitive object control frames.** (a) A *Prism* with control frame. (b) A *Frustum Cone* with control frame. (c) An *Ellipsoid* with control frame. (d) The control frame of object (c).

6.4 **Colors used on primitive object control frames.** (a) A *Frustum Pyramid* with control frame. (c) An *Ellipsoid* with control frame. (d) The control frame of object (b).

6.5 **Control point auto snap features.** (a)-(b) *Column* object control point auto snap. (c)-(d) *Ellipsoid* object control point auto snap.

6.6 **Model precision adjustment.** (a) Adjust the edge number of a *Polygon Section Prism* from 4 to 8. (b) Adjust the smoothness precision of an *Ellipse Section Cone* from Fine to Coarse. (c) Adjust the smoothness precision of an *Ellipsoid* from Moderate to Fine.

6.7 **Primitive model creation.** (a) Create a 3-edge *Prism* with regular triangle section. (b) Create a coarse precision *Frustum Cone* with ellipse section. (c) Create a partly symmetric *Ellipsoid* with moderate precision.

6.8 Eight *Ellipsoid Pieces* are stitched together to construct a *Primitive Ellipsoid*. (a) A typical convex *Primitive Ellipsoid*. (b) An *Ellipsoid Piece* of (a) showed in blue. (c) Three half-axes control an *Ellipsoid Piece*. $C^1$ or $G^1$ continuity maintains at the intersections of those eight ellipsoid pieces for this convex ellipsoid.
6.9 **Column object selection.** The selection loop of a column object is showed in this figure. (a) shows an unselected *Frustum Cone* and the entire object is selected in (b). The bottom and top surfaces are selected in (c) and (d) respectively. .......................................................... 52

6.10 **Ellipsoid object selection.** The selection loop of the ellipsoid objects is showed in this figure. An *Ellipsoid* is totally selected in (a). In (b), the horizontal control frame of this ellipsoid is selected, while only the vertical half-axes are selected in (c). The bottom control point and the upper control point are selected in (d) and (e) respectively. .......................................................... 52

6.11 **Column object surface movement in 2D and 3D workspace.** In (a)-(b), the top surface of a transformed *Frustum Pyramid* is horizontally moved in 2D workspace by aligning a control point of the top surface to a control point of the bottom surface. This operation changes a skew side face to a vertical one. In (c)-(d), the bottom and top surfaces are vertically moved in 3D workspace. This operation makes an ellipse-section *Frustum Cone* switch its two horizontal surfaces up side down. The height and elevation of this column object are also changed. .......................................................... 53

6.12 **Column object surface transformation.** The column object mutual-conversion is showed in (a)-(c). A polygon-section *Prism* (a) is transformed to a *Frustum Pyramid* (b), and sequently changed to a *Pyramid* (c). In (d)-(f), by modifying the top surface of an ellipse-section *Cylinder* (d), we obtain a round top surface *Frustum Cone* (e) and a funny column model (f). .......................................................... 53

6.13 **Ellipsoid object horizontal control frame elevation alteration.** This figure shows how to change a convex *ellipsoid* to a concave *ellipsoid*. .......................................................... 54

6.14 **Ellipsoid object vertical axis length adjustment.** This figure shows how to symmetrically change the vertical axis of an *Ellipsoid* object. .......................................................... 55

6.15 **Ellipsoid object bottom/top control point movement.** This figure shows that by vertically moving the top control point of an ellipsoid object, we can change an ellipsoid (a) to a half ellipsoid (c), and finally make it a half shell (d). .......................................................... 55
6.16 **Primitive model 2D center modification.** After selecting the top surface of a square prism, we (a) move the mouse cursor to the center control point, and (b) press-down-move mouse into the 1st quadrant, or (c) press-down-move mouse into the 2nd quadrant, or (d) press-down-move mouse into the 3rd quadrant, or (e) press-down-move mouse into the 4th quadrant.

6.17 **Column object horizontal surface selection auto-switch.** An ellipse-section Frustum Cone is showed in (a) with the top horizontal surface selected. (b) depicts the critical plane when two horizontal surfaces superpose together. In (c), the originally selected top surface is automatically switched to bottom surface after it crosses the critical plane.

6.18 **See-through and Footprint observation features for Ellipsoid objects.** These features are illustrated by an ellipsoid object in this figure. The 2D selected ellipsoid, its see-through result and footprint are showed in (a). The 3D corresponding views are showed in (b).

6.19 **An example of modifying Ellipsoid objects based on half-axis observation.** (a)-(j) depict the modification processing of gradually changing a non-symmetric ellipsoid to a uniform sphere.

7.1 **Some simple models created using our system.** (a) depicts some typical rudimental shapes we can obtain in the system. An imaginary flower and a goblet are showed in (b) and (c) respectively.

7.2 **Two examples of modifying simple architectural models for conceptual designs.** These coarse yet impressive models can clearly show architects’ initial imagination towards some specific projects, and can be used as the basic of performing detailed designs.

7.3 **A conceptual design of a museum finished using our system.** (a) shows the 3D translucent model while the 2D top view is showed in (b). (c) depicts a detailed perspective view of the entrance. Using the modification functions provided by our system, creating typical slope roof is an easy job in the modeling process. Different observation modes can let architects control their designs easily.
7.4 A courtyard landscape conceptual design obtained using our prototype. (a) illustrates the integrated design/draw environment provided by our system, while (b) shows the 3D result. After obtaining the models in our system, we exported them out to an "obj" file and loaded it into 3ds MAX. (c) and (b) depict the top view and perspective scene rendered by 3ds MAX. This example shows the file quality of the exported models is good enough for transferring 3D data into industrial modeling tools.

7.5 A London Tower Bridge model created using our prototype based on a photo. (a) and (b) illustrate the 2D and 3D design results, while (c) shows an perspective rendering obtained in 3ds MAX. Here the Extrusion and Primitive models are combined together to create complex architectural entities.

7.6 An model of the new St. Anne's Church, Bukit Mertajam, Malaysia. (a) depicts the 3D model obtained using our system and (b) illustrates the crisscross top view. (c) shows a perspective rendered by 3ds MAX. Almost all the geometric shapes in this example are Primitive objects. This shows sometimes the Primitive method is a very efficient way for creating architectural models.
Chapter 1

Introduction

Architecture is the art and science of designing buildings and structures. The role of the architect, although constantly evolving, has been centered to the design and implementation of the environments in which we live. Although architectures, as thinking products of the architects, are almost exclusively 3D entities, their initial conception has always come from the architects’ drawing papers, typically through the use of freehand sketches by a pencil; the 2D drawing papers are the architects’ design environment.

Traditionally, architects draw perspective pictures to evaluate, refine or exhibit their designs. This is a time-consuming task and not every architect is an expert on perspective drawing. Even if the drawing comes from a skilled architect or artist, the problem of perspective distortion typically occurs in every hand-drawn picture. With the rapid advances in computer-aided design, computer graphics, and human-computer interaction techniques, more and more computer-based systems have appeared in the field of architectural design and modeling. These software products aim to extend the architects’ ability to express, to communicate, and to experiment, utilizing their creativity in a fuller extent.

There is little doubt that architectural design in the future needs to seamlessly integrate the architects into the digital modeling work-flow, the realization of such a goal however presents many challenges. Although much progress has been made at the rendering stage, architectural modeling, particularly at the early stages of design, represents one of the most difficult challenges for the interactive graphics software industry. Currently, available architectural design software is difficult to use at the preliminary design phase. Most offerings are derived from two-dimensional drafting systems requiring precise input. What is needed is a ‘back-of-the-envelope’ environment, with the ability to sketch, doodle, and erase, to
CHAPTER 1. INTRODUCTION

proceed in an iterative, non-linear fashion, and to perform these operations in a full three-dimensional domain. Since design is an iterative process and it is frequently necessary to work conceptually in two dimensions as well as three, it is important to be able to migrate freely from one domain to another. It is just as important to be able to extract plans, sections, and elevations from 3D sketches and models, as it is the other way around.

The above assessment has been echoed by other researchers, e.g., [4, 20]. Most design studios nowadays still use AutoCAD to finish 2D drawings and 3ds MAX to create 3D models. These industry standard software offer powerful functions that can be used for architectural design, where precise geometric parameters serve to connect the user with the digital models they create. Specifically, the user interface adopted by these software products forces the user to frequently provide parameters, such as lengths, radius, etc., during the design process. We can call these modeling tools parameterized software. Such a mode of interaction has proven to be quite useful for accurate and detailed designs. However the many parameters, which can be rather complex to learn, memorize, and manipulate, are rendering the system less suitable to use in the early design stage, which is iterative, non-linear, and imprecise in nature [9]. In contrast, when software products can let the user freely create various geometric entities without having to enter parameters, designers can focus more on the design itself. We roughly call these software behaviorized software since they conform more to the drawing behavior of a designer during the creative process.

1.1 Motivations and Applications

The key point is that architects are not necessarily experts in digital 3D modeling. The interaction and design functionality offered by the powerful general-purpose modeling software do not conform to the traditional habits of an architect during conceptual design. As a result, in practice, many architects commonly send 2D drawings to other artists or 3ds MAX operators to finish the 3D models. A great deal of resources such as time, money, and human efforts can be consumed on communication between the architects and the modelers and in general, it is impossible for the latter to totally understand the desires of the architects without discrepancy.

In this project, we aim to address the issues outlined above and propose a prototype sketch-based architectural design system, with simplicity and design habits of the architects
CHAPTER 1. INTRODUCTION

Figure 1.1: A model of Master architect Frank Lloyd Wright’s “Fallingwater Villa” created by our prototype system. (a) Sketching in 2D. (b) Assembly and extrusion. (c) Modification (colored polygons). (d) Final product. The figure shows the work-flow of our system and illustrates the system functionality of sketching, assembly, and modification.

In mind. Our entire system design philosophy is dictated by what we refer to as the traditional architectural design habits (Chapter 3). We implement the migration between the 2D and 3D design domains via the notion of descriptive geometry (Section 3.1), a graphical technique allowing for the creation of 3D virtual scenes on a 2D plane via the manipulation of heights [2]. We are dedicated to provide a design environment conforming to practical architectural design workflow (Section 3.2).

In the context of architectural design, our goal is to aid an architect in the early design stage, the stage of scheme conceiving, consisting of sketching, comparing, adjusting, visualizing, rejecting, accepting, and communication. Figure 1.1 illustrates several aspects of our system and shows a sophisticated model obtained with functionality implemented by the current prototype.

Instead of relying on a computer algorithm to “guess” what an architect desires, e.g., as done in works on sketch interpretation [12] and 3D model reconstruction from sketches [4, 16], where surprising or distracting results may hinder the design process, our system grants full control to the architect so as to utilize his experience, creativity, and ability to balance the multiple aspects of architectural design. Such a process is made easy and intuitive via our insistence to respect the traditional architectural design habits.

Some CAGD software products, such as AutoCAD, 3ds MAX and SketchUp, etc., are widely used in architecture design. They are almost industry standard software for architecture design. Lots of designers use AutoCAD to finish 2D drawings and use 3ds MAX to create 3D models. These software have been developed for many years and all kinds of
powerful functions relating to design are perfectly provided, especially when detailed designs are wanted.

In the design environment provided by these software products, mouse and keyboard are devices for users to interact with computers. Various geometry parameters are the interim connection between users and computers and they are quite useful on doing accurate and detailed designs. But at the same time, these parameters are somewhat complex for designers to create 3D models. The huge number of parameters become a barrier in the process of interaction between designers and computers.

In practical application, many architects only use AutoCAD to finish 2D drawings. When 3D models are needed, architects commonly send 2D drawings to other computer artists or 3ds MAX operators, typically who are not architects and working in other companies or studios, to finish 3D models using 3ds MAX. Lots of time, money and energy have been consumed on communication between architects and artists. In general, it is impossible that architects' design ideas can be totally understood by artists without discrepancy.

The reason for this low efficiency work is that the skills of using 3D modeling software is quite beyond architects' ability. To use 3ds MAX expertly, an operator must cost adequate time to get used to its application environment and memorize lots of parameters. It is hard for architects to invest time and energy to become a skilled 3ds MAX operator, especially when they have many design projects in hands. Meanwhile, it is not possible for designers to use mouses and keyboards as freely as using pencils and papers, because these parameterized software are mostly designed based on scientific principles but not on traditional design habits. So, there is a gap between designers and 3D modeling software when conceptual designs are needed. Some kinds of software which can be used as freely as using pencils and papers are needed to improve architects' design efficiency and quality.

1.2 System Features

In this project, we present a prototype modeling system, which is simple, intuitive, and efficient for generating geometrical models. The techniques used in this system, such as curve design and modeling, Delaunay triangulation, etc., are not novel; they are mostly well established techniques applied to our system. We view our contribution as the development of a practical design and modeling tool for the architects in assisting them during the early conceptual design stage. Using this modeling tool, architects can easily create 3D
models with their traditional design habits. Our modeling systems has a number of desirable features, which we list below.

- **Integrated design/draw environment**
  
  All those practically-being-used powerful software products mentioned here, e.g. AutoCAD, 3ds MAX and SketchUp, etc., have their own features. But there is a common shortage existing in all of them. That is the lack of an integrated design/draw environment based on traditional design habits. Our work is dedicated to provide such an integrated design/draw environment.

  Here “design” means users can draw sketches on a computer screen with the same experience of drawing sketches on a paper with a pencil. Although SketchUp claims that it emulates the feel and freedom of working with pencils and papers, unfortunately, it has no sketch mode and uses a mouse to do everything. “Draw” means users can create 3D models based on their conceptual designs, the sketches they are drawing on the screen, only with a pencil-like stylus. Sketch-based Modeling software supporting traditional design habits should provide such an integrated design/draw environment. This kind of behaviorized software should naturally conform to human drawing behavior mode.

- **A focus on interaction**

  The core idea of this integrated design/draw environment is interaction. Suppose an architect is doing a practical design, can this design be done only by drawing some sketches in 2D or 3D workspace and then an original concrete architectural model can be automatically given back by a computer? Obviously, the answer is in the negative. The process of architecture design is far complicated than this description.

  Good architectural designs are based on architects' study, experience, genius and ability of keeping equilibrium on multiple design aspects. These abilities can not be substituted by computers. Computers are not needed to have the ability to "guess" what a model looks like. Surprising or distracting results possibly created by a computer will not be expected by designers. This can only interrupt designers' thinking progress and then hinder design efficiency.

  From a blank paper to a fantastic architectural scheme, software should be handy tools, just like pencils and papers, to assist architects to do various operations related
to their designs. Every detail of a 3D model should be under control. Designers should be allowed to do what they want to do with simple yet efficient operations and get reasonable results based on their thoughts. This is the soul of CAD and our project is exactly based on this interactive idea.

- **Potential of utilizing features of Tablet PC system**

Mouse and keyboard should be abandoned in behaviorized software environment. Pencil is a tool for us to draw pictures on flat surfaces. Obviously, it is much easier to use than a mouse when we draw sketches. Pencil is simple, easy to grasp and can be tightly fixed in hands and, it can easily draw all kinds of lines in various directions. In a word, pencil is a simple yet efficient tool for designers to express what they want to express. These features explain the reason why designers do not like to use mouses to draw sketches. Compared with pencils, mouses are strange-shaped, hard to be tightly fixed in hands and, they can not easily draw all kinds of lines in various directions. Mouse is not an efficient tool for designers to express what they want to express when wonderful ideas surging in their minds. In other words, mouse is good at pointing but not at freely drawing.

So, we need a set of devices which can be used like pencils and papers. Tablet PC is an idea choice of hardware interface. A Tablet PC system dose have an input pencil-like stylus which can be used to freely draw sketches on the flat computer screen like a real pencil. The screen of a Tablet PC can also be flatly put on a designer's desktop like papers. In this project, for the general purpose of testing our prototype system, we still use a mouse to simulate the stylus. Various operations can be done only with a mouse and no action is based on keyboards. All the functions activated by mouse buttons are designed to have mapping ability to specific function keys on a stylus. That means we have the potential of finishing all the interactive operations only with a stylus. Section 4.2 gives out details about this mapping mechanism.

- **Based on traditional design habits**

Parameters used in *AutoCAD* and *3ds MAX* control every geometry detail of a model, so the model can be created very accurately. But in many occasions, especially when a designer is conceiving an architectural scheme, generally he dose not care about the accurate geometry parameters of every line segment he is drawing or dimension values
of the model he is creating. In the stage of conceptual design, what a designer cares about is what the model roughly looks like and how to modify it into a perfect one just as what is being conceived in his mind. When a designer uses a computer to create a new model not existing in this world before, he will feel upset on dealing with too much less important information of his design.

So, in the stage of conceiving an architectural scheme, what an architect wants most is a correspondingly coarse yet expressive model. What he wants most to do is, to create and modify the model easily. This should be done in an integrated design/draw environment of the behaviorized software. All the operations designed in our prototype system are based on traditional architectural design habits and dedicated to conform to human drawing behavior mode. Here “human drawing behavior mode” refers to “our natural habits or tendency of drawing”. This is a basic and important concept of interaction in our system. It comes out from the traditional design habits of architects. Sketches are allowed to be drawn anywhere and anytime. Designers can extract valuable thoughts from these sketches and then easily create 3D models. It provides the possibility for designers to do what they want to do as naturally as possible.

1.3 Thesis Organization

The rest of this project report is organized as follows. We begin by discussing some related works in Chapter 2. In Chapter 3, we discuss some issues related to architectural design and describe a simple work flow of it. Then in Chapter 4, we provide a system overview. In-depth descriptions of the various system functionality and interaction paradigms appear in Chapter 5 and 6. The design examples section, Chapter 7, provides illustrative examples and exhibits several architectural models obtained using this system. Finally in Chapter 8, we conclude and mention our plans to improve upon the current prototype.
Chapter 2

Related Works

2.1 Computer-aided design (CAD) modeling software used in architectural design industry

Here we list out some software which are typically used in architectural design. From 2D to 3D design, some of them provide perfect functions in the stage of technical drawing. But in the stage of scheme conceiving, i.e. conceptual design, these software are somewhat bulky. Some software focus on conceptual design, but they don’t provide sketch-based design environment. This limits designer’s ability of creation and dose not conform to architects’ design habits.

- **AutoCAD** is a suite of CAD software products for 2D and 3D design released by Autodesk, Inc. *AutoCAD* includes a full set of basic solid modeling and 3D tools, but lacks some of the more advanced capabilities of solid modeling applications. *AutoCAD* is a vector graphics drawing program. It uses primitive entities - such as lines, polylines, circles, arcs, and text - as the foundation for more complex objects. *AutoCAD* is most popularly used in architecture design, especially in 2D drawing. A "sketch" command is provided by *AutoCAD*, but its function is quite limited on recording mouse cursor position.

- **3ds Max** is a full-featured 3D graphics application developed by Autodesk Media and Entertainment. It has strong modeling capabilities, a flexible plug-in architecture and a long heritage on the Microsoft Windows platform. It is mostly used by video
game developers, TV commercial studios and architectural visualization studios. 3ds MAX provides powerful modeling functions such as Polygon modeling, NURBS or Nonuniform rational B-Spline modeling, etc. But it does not provide a sketch mode.

- **Google SketchUp** allow designers to draw with the way they want by emulating the feel and freedom of working with pen and paper in a simple yet elegant interface. SketchUp is a powerful yet easy-to-learn 3D software tool that combines a simple, yet robust tool-set with an intelligent drawing system that streamlines and simplifies 3D design. From simple to complex, conceptual to realistic, SketchUp enables users to build and modify 3D models quickly and easily. But it does not provide a real sketch mode either. Besides drawing simple arcs, the system can only record freehand mouse movement and the constructed line stripes are not recognized as curves, thus cannot provide users smooth curves they want and makes further curve modification impossible.

### 2.2 Academic research on sketch-based modeling

In the academic community, sketch-based modeling and interfaces have received a great deal of attention lately. A variety of tasks for which the use of sketches conforms to our behavior mode have been considered, these include product [4, 23], animation control [22, 27], garment design [29], and shape analysis such as segmentation [31], among others.

In this chapter, we focus on some related techniques and technologies needed to enable sketch-based interfaces. Based on a tentative classification of the researches aiming at the sketch-based geometric modeling issues, we give some references of the work in these relevant areas, and also discuss some related works developed in the past that we have made use of in our work in the following 6 categories.

#### 2.2.1 General topics on sketch-based modelling

These works described in this section bring us a comprehensive concept about the problem of sketch-based geometric modeling. From various well-known systems in this research area, such as *SKETCH* [32], *Teddy* [10] and *Quick-sketch* [5], etc., their works present the general definition and approaches to the sketch-based modeling issue. In the survey paper [4], the
CHAPTER 2. RELATED WORKS

background and evolution of three-dimensional reconstruction of line drawing over the last thirty years is discussed.

Gooch’s book [7] provides an overview of the published research on non-photorealistic rendering. But non-photorealistic rendering is not only about the images. NPR has also had an effect on geometric modeling. In the NPR section of this book, Sketching to Create Models, Gooch illustrates three modelling system to show us some typical research directions and applications using sketch-based techniques. SKETCH system [32] combines NPR with gesture recognition to create three-dimensional geometric models. Teddy system [10] allows a user to draw the silhouette of an object, which the system uses to create a three-dimensional polygon mesh. Another system mentioned in this section allows users to draw their own world and move around in it.

In the survey paper [4], Ana Piquer et al. discussed the evolution of geometry reconstruction. The author divided geometry reconstruction problem into two categories distinguished between single view and multiple view approaches. The evolution process, from recovering know-how stored in engineering drawings to sketch-based modeling application in conceptual design stage, has been discussed in this paper. In the evolution of geometrical reconstruction the emphasis has moved from geometry to perception. In this domain, psychology and perception rules play the most important role. For the current challenge of geometry reconstruction problem, “perception” gets to be more and more important in the processing of geometric modeling problem as an ever-increasing tendency.

Lynn Eggli et al. [5] presented a basic and foundational paper talking about sketch-based modeling approach and a tool called “Quick-sketch” is provided along with it. Although it is a relatively old paper, many techniques applied in this paper are still useful in today’s sketch-based modeling systems. Based on people’s drawing habits, this paper presents us with some intuitive ways to do sketch drawing, recognition and modification. From an interaction point of view, various gesture-based interfaces have been developed for 3D shape creation and modification. A drawback in this sketch-based modeling system is that it relies on a constraint mechanism to do recognition and this should be explicitly specified by the user.

SKETCH system is a typical gesture-based user interfaces, where users create 3D objects using a set of predefined gestures. Robert C. Zeleznik et al. [32] attempt to combine the advantages of traditional paper-pencil interface and computer 3D modeling techniques in order to create an environment for rapidly conceptualizing and editing approximate 3D
CHAPTER 2. RELATED WORKS

It uses simple NPR and a purely gestural interface based on simplified line drawings of primitives that allows all operations to be specified within the 3D world. The main disadvantage of this system is that the gesture is not easy to remember and can sometimes be distracting.

2.2.2 Using sketch to create free-form objects

In this section, some sketch-based modeling systems for creating free-form 3D geometric models, such as Teddy [10], ShapeShop [24], FiberMesh [18], etc., are presented. Sketch-based free-form 3D geometric modeling technique is a well-studied branch in this modeling field. Given a simple closed stroke, a free-form modeling system can generate a shape matching this contour. Closed triangle meshes can be created by inflating user-sketched 2D contours using the chordal axis of the 2D polygon. Users can add details by editing the mesh with operations like extrusion, cutting, bending, and drawing on the mesh.

Takeo Igarashi et al.'s Teddy [10] is a modeling system that allows the creation of certain free-form objects with a sketching interface. By drawing several 2D freeform strokes interactively on the screen, the system automatically constructs plausible 3D polygonal surfaces. This system allows users to sketch the silhouettes of objects in 2D space and then inflate them into rotund 3D polygonal meshes matching those contours. Users can add details by editing the mesh with operations such as extrusion, cutting, bending, and drawing on the mesh.

ShapeShop [24] of R. Schmidt et al. is a sketch-based modeling system falling into freeform modeling tools. Along with the “Blobby inflation” technique used in this system, the “Sweep surface”, sometimes called the “Extrusion”, is also implemented in it. This system is correspondingly mature in creating free-form models and the illustrating pictures are impressive. One of its drawback is that it can not preserve sharp features in the meeting parts of objects. In 2D modification, this system uses “variational implicit curve” to represent sketches. This brings us good features in the smoothness of the contours, but the sharp features we need in contours are lost at the same time. Even though, this system definitely provides us with a useful framework towards practical application.

SmoothSketch [14] presented by Olga A. Karpenko et al. can create geometry models by inferring plausible 3D free-form shapes from visible-contour sketches. After the user draws the visible contours of a shape, including cusps and T-junctions, SmoothSketch infers the hidden contours, including hidden cusps, and then creates a fairly smooth 3D shape
matching those contours. Some good features of this system are it can handle objects with complex holes and there is no constraint that a user’s sketch is need to be a simple closed curve. One drawback of this system that the contour-completion approach is local instead of global prevents it from working universally.

Andrew Nealen et al.’s FiberMesh [18] makes a big stride on free-form modeling. Control curves are not only used to generate a rough 3D model, but also as handles to modify the model in real-time. Observably distinguished from other from-form modeling system, FiberMesh can keep smooth and sharp features on the model simultaneously to allow models to contain more details. Control curves can be directly added on the surface of the model and the arbitrary topology property of these curves greatly frees user’s hands when manipulating the model. Smooth and sharp curves can be mutually changed and thus brings the model interesting geometrical presentation by incurring a gradual transformation between smooth and sharp features.

Plushie is a free-form modeling system used to design plush or balloon toys focusing on practical use, presented by Yuki Mori et al [17]. This system can generate cloth patterns for sewing and the shape of the objective toy conforms to the virtually design one by applying simple physical simulation. Seam lines can be added on the model interactively and this makes material arrangement possible. Patches connected to each other using connectors and numbers shows that this system can be used as a practical sewing tool.

2.2.3 Create featured objects based on sketch

Some target objects have their intrinsic features embedded in themselves, e.g., the shape of a flower and the spacial relationship of the leaves on a plant. These features can be taken as the ground knowledge towards those special objects with obvious properties and thus gives the modeling process apparent focuses on the targets. The systems presented in the following papers in this section can create impressive models such as leaf, flower, skirt and sword, etc. User’s sketches play the important role of the hints and parameters to make the model generation process more intuitive.

Joseph Jacob Cherlin et al. [3] present a system which implements sketch-based modeling using interpolating parametric surfaces. Unlike the usual blobby aspect of free-form 3D models, this system allows an efficient modification of the local or global shape, and the generation of more complex models in few sketches by editing the profile curve instead of using a skeleton to inflate the object volume. Inspired by the traditional illustration strategy
CHAPTER 2. RELATED WORKS

for depicting 3D forms, this system is suitable for creating some specific features, such as creases, sharp corners, facets and bumps, on the model. A drawback of this system is each part object must be manually positioned to give the impression of a solid 3D model. While complex models can be created, the requisite manual positioning is very time consuming.

Fabricio Anastacio et al. [1] propose a sketch-based approach to create specific single-compound 3D plant models. Following the traditional botanical illustration technique of concept sketching, plant features such as phyllotactic patterns are used to direct the creation of the plant model. Sketches can be modified to refine the existing model using linear interpolation technique. For the reason that the features of the target plant models are definitely defined as underlying knowledge based on botanical research, the sketch-based modeling and modification interface provides users an intuitive and effective manipulation mechanism. A drawback of this system is it is only dedicated to a single type of plant arrangement.

T. Ijiri et al. [11] present a sketch-based modeling system to intuitively create 3D flower models. This system focuses on particular flower features and employs a collection of 2D sketches to guide the process of modeling. It uses a top-down modeling approach to catch the ability of simultaneously modeling details and controlling global information. Using a gradually developing processing, the flower models can be created from coarse sketches along with drawing additional decorative strokes. Unlike other rule-based approaches, this system needs the user to control all details of the target model. This brings a drawback to this system by limiting the creation of huge plants.

2.2.4 Reconstruct objects using sketch

Given 2D sketches representing 3D objects in a plane, the automatic 3D geometric model reconstruction process is an intuitive way to create objects; and thus makes model reconstruction an interesting research direction in the sketch-based geometric modeling field. In this section, different systems presented in those papers give us some concrete examples to show that sketch-based geometric reconstruction process can bring us very useful modeling tools.

Chen Yang et al. [30] explore an idea of using pre-defined 2D sketch templates to recognize and convert users' sketches into 3D shapes. A graph hierarchical representation for sketches and templates is used in it and matching is done using curve feature vectors coupled with a scoring function. Finally a 3D model is created based on the selected template
using information extracted from the sketch to parameterize the process. The recognition and 3D geometry construction algorithms make this approach suitable to a limited number of objects with relatively simple geometry.

M. Masry et al. [16] present a system in which the user can quickly build a CAD model via sketching and immediately apply finite element analysis to the model. This system can progressively reconstruct an object's 2D sketch representation, including straight lines and planar curves, into 3D objects by implementing two optimization-based reconstruction algorithms. Using finite element analysis, the structural properties of the created 3D objects can be examined and displayed in an intuitive way.

D. C. Ku et al. [15] present a sketch-based interface to reconstruct 3D polyhedron. Freehand sketch can be converted to a vertex-edge graph from 2D tidy-up processes and then by using 3D geometry approximation and hidden topology determination, the system can create reconstructed objects after transformation. Sketches can be continuously added on the existed object and thus makes an incremental modeling process possible.

2.2.5 Modify objects using sketches as guidance

The researches done in this section illustrate us some useful applications of using sketches as guidance. Sometimes we want to add subtle features to existing models, and sometimes we try to modify models to better ones. Sketches can be used to do these jobs by making themselves guidance to the changes. These changes can not only be the difference between local geometric details of models, but also the transformation operations such as translation, rotation and non-uniform scaling. Some application of sketch-based approach can even make the outdoor point cloud segmentation possible.

L. Olsen et al. [21] present a sketch-based approach to do shape augmentation on geometric meshes. It allows users to directly draw sketches on the surfaces of existing 3D shapes and the details such as sharp features, convex and concave regions, etc., can be added on the models. Adaptive subdivision technique is used in this paper to refine the mesh within the vicinity of sketches and thus the goal of mesh augmentation and feature creation is achieved by displacing vertices in the mesh. For the reason that the most computationally expensive operation, adaptive subdivision, is done based on the complexity of the features instead of the mesh itself, typical meshes can be augmented at interactive rate and thus makes this system have a practical use ability.
Xiaoru Yuan et al. [31] use a sketch-based interface to segment scanned outdoor 3D point-based models. Guided by different strokes the user places on the object and background, the system segments out the marked objects. A two-pass process, first performs 2D image segmentation in camera projection plane guided by strokes and then uses the result to segment 3D objects, is employed to finish the objective segmentation. Posed as a 2D graph cuts problem, global optimal pixel labeling of an object and its background can be used as a combinatorial optimization technique.

Aaron Severn et al. [25] present an approach to perform transformations in a modelling system using a single stroke as guidance. Transformation problem can be solved easier in their approach because user’s intentions towards transformation can be extracted out from strokes which contain more information than a mouse click. To eliminate ambiguity, 2D strokes are constrained to be U-shaped. Using principle component analysis (PCA), the major and minor axes of a U-shaped stroke can be computed and transformation information including translation, rotation and non-uniform scaling is then stored in them.

Levent Burak Kara et al. [13] present an intuitive, sketch-based interface to directly create and edit free-form curves and surfaces. Start from an initial template surface model, users can create and refine objective 3D models by modify sketch curves on the model surface as guidance. By using this template based curve modeling and editing technique, this system constrains sketch curves on the existing geometry. Free-form 3D curves with varying depth coordinates can be created so long as the two ends of the curve can be anchored in 3D using existing primitives in the scene.

2.2.6 Sketch-based architectural modelling

Traditionally, architects draw sketches on papers to aid them in conceiving conceptual designs. Sketch-based 3D modeling research is obviously an interesting and useful approach to extend an architect’s drawing board to virtual reality. From projecting designer’s sketches onto a sphere to creating solid architectural models using sketches as reference lines and gesture commands, lots of sketch-based modeling researches try to intuitively simulate traditional design method and environment for architects.

Osama Tolba et al. [28] use Non-photorealistic rendering to simulate architecture drawing and it can preserve hand-drawn strokes via reprojection. It lets a user draw a scene with 2D strokes and then view it from several new locations as if a 3D scene were created from it. This is done by projecting the 2D strokes on the sphere with the center at the eye point and
then viewing them in perspective. However, the paper-machine-iteration work flow provided in this paper is not suitable in practical use. Although this stroke-based approaches reduce the limitations on possible 3D lines and curves, but it is limited to illustration since cannot really reconstruct a full 3D object.

Roland Juchmes et al. [12] present a sketch-based multi-agent system to interpret architecture scratch in the early design stage. Different types of agents are used in this system to capture and interpret the meaning of sketch drawings and these agents can communicate with each other to make feasible and pertinent decisions. However, although this agent mechanism has some advantages on recognizing architecture entities, this system is not a modeling tool towards creating 3D architecture models. This system focuses on understanding sketches but not on model creation and this drawback greatly limits its practical application.

SMARTPAPER [26] presented by Amit Shesh et al. is a sketch-based modeling system that can be used to create and manipulate planar objects. A user study of testing it by a group of students from the Department of Architecture shows its ability to allow sketching freely without much learning. It can interpret 2D line drawings to closed 3D polygonal objects and the approach of sketching over a 3D view using a predefined gesture syntax where geometric aspects of gestures determine numerical parameters of the objects is adopted by this system. However, SMARTPAPER is limited to straight-line input and doesn't work with arbitrary curves as primitives. There is no discussion of wrongly drawn line correction and hence it is assumed that the system is limited to interpret sketches with correct strokes.

Ji-Young Oh et al.'s SESAME [19, 20] incorporates 2D drawing and extrusion abilities to allow users to create shapes such as architecture entities with intuitive ways. Users can quickly sketch the outline of a shape, extrude it to create a 3D model and then carve it. A gravitational hierarchy grouping technique improving manipulation is used in this system and thus prompts efficiency significantly. A drawback of SESAME is its use of keyboard and mouse to do all the modeling operations and thus can not provide users a natural enough interaction interface. Using a more intellectual way instead of explicitly selecting operation commands from control panel to let system perceive user's intention during the process of interaction can be an improvement direction.
Chapter 3

Design Habits

We can roughly divide the architectural design process into two stages, one is conceptual design and another one is technical drawing. Traditionally, architects are trained to use the paper and pencil to perform conceptual design, and use modern CAGD software, such as AutoCAD, etc., to finish final drawings. The whole design progress is a process of working from physical papers to virtual ones, although architects often go back and force between these two status in reality.

For an architect, the second important stage, technical drawing, is correspondingly simpler than conceptual design. It only requires architects to provide technical drawings satisfying building codes and economical requests. This design ability is not too hard for an architect to hold in practical work. Typically, architects use AutoCAD to finish this drawing work.

On the other hand, the most important stage is the first one, conceptual design, because it is a process of creation. In this stage, architects draw numerous sketches on tracing papers with pencils. This is a stage of conceiving, sketching, comparing, rejecting and accepting. It is especially important to the quality of a design. Those words, such as inspiration, originality, spark, etc., are typically used to describe this stage and it is also the key reason to impress clients and the public.

What we called behaviorized software is designed to be used in this conceptual design stage. Behaviorized software should has the ability to help a designer to conceive and put forward a scheme never existing in this world before, as easily and intuitively as possible. Unlike parameterized software such as AutoCAD and 3ds MAX, the soul controlling everything in behaviorized software is not the parameter, but the traditional architectural design
habits.

In the following subsections, we first describe a useful drawing technique used by architects, the Descriptive Geometry. Then we discuss a simple architectural design work flow implemented in real design.

### 3.1 Descriptive geometry

Architects are trained to perform architectural design via 2D projection drawings, using the method of *Descriptive Geometry*. Descriptive Geometry was proposed and consummated by Gaspard Monge (1746-1818), a French mathematician and inventor of Descriptive Geometry. It is a graphical technique used to create 3D virtual spaces on a 2D plane. Generally, an undergraduate student major in architectural design will take the Descriptive Geometry course in his/her first or second year and this 2D-3D thinking and drawing technique will be used in his/her entire design career.

Descriptive Geometry is a graphical technique which creates three-dimensional virtual space on a two-dimensional plane. Monge’s protocols allow an imaginary object to be drawn on a 2D plane in such a way that it may be 3D modeled. All geometric aspects of the imaginary object are accounted for in true-size/to-scale shape, and can be imaged as seen from any position in space. All images are represented on a 2D drawing surface. Descriptive Geometry uses the image-creating technique of imaginary, parallel projectors emanating from an imaginary object and intersecting an imaginary plane of projection at right angles. The cumulative points of intersections create the desired image.

To an architect, a basic 3D geometric model is created by giving a height to a 2D polygon and then modifying it. A complex model can be looked upon as a combination of these solid geometric models which have been placed at appropriate spatial positions with desired orientations. Thus the process of creating architectural models is nothing but establishing shapes of base polygons and giving heights to them, called *Extrusion*, combined with appropriate modifications. Our system exactly conforms to this design habits of architects by utilizing extrusion as the core 3D construction method.
3.2 Design work flow

Now we ask ourself a question: How to make our behaviorized software conform to traditional design habits? To answer this question, let us ask another more practical question: How does an architect finish a 3D model on a real paper with a real pencil? We try to answer these questions via a simple discussion about the work flow of an imaginary architectural design.

Here we will take a simple building design for an instance. First we show the final 3D model in Figure 3.1 (a) to give out a general impression of what we will discuss about below. We can imagine that this simple building stands on a site as what we show in Figure 3.1 (b). This is the top view of this building and it is drawn in this site map. We can find that this building consists of three parts: two rectangles and one quarter circle (sector).

We can state that if we have gotten Figure 3.1 (b), we can easily draw out Figure 3.1 (a) using *AutoCAD* or *3ds MAX*. That’s right. But the problem is: Before a designer can draw a picture like Figure 3.1 (b), perhaps he has drawn more than 10 sketches like what we show in Figure 3.2 (a)(b)(c). The reason is very simple: At the very beginning, the designer only got a blank site map showed in Figure 3.1 (c) which was given to him by a client in advance.

So, we now have discovered the work flow of designing this simple building: A professional architect finishes this job from Step (a) to Step (d) showed in Figure 3.3. *AutoCAD* or *3ds MAX* only finishes the job from step-3 to step-4 of Figure 3.3. Traditionally, other important conceptual design work from step-1 to step-2 of Figure 3.3, must be done by an architect with paper and pencil. In a practical design, the design done from step-1 to step-2 is the soul of a creation process. So much work, such as originality, comparison, modification, adjustment, etc., should be done in this process. Unfortunately, software such as *AutoCAD* and *3ds MAX* can not perform the work from step-1 to step-2. These software are tools for designers to finish final drawings on the precondition of having thought every detail well in advance. The important conceptual design stage can not be easily finished by using parameterized software. It should be done by using behaviorized software conforming to architects’ traditional design habits.

What we have discussed above is only one aspect of traditional architectural design habits imbedded in behaviorized software. Behaviorized software should let a designer draw all kinds of sketch lines in the design environment to aid the designer in conceiving and
finishing the work in the whole conceptual design progress. Behaviorized software must have the ability of extracting useful information in the sketch lines to construct the final 3D model. In a word, we should set a digital sketch mode anywhere we want it in the behaviorized software environment. This thinking method can be applied to every detail of behaviorized software conforming to architectural design habits.
CHAPTER 3. DESIGN HABITS

Figure 3.1: Architectural drawings. (a) A simple building model. (b) Top view of the building. (c) Site map.

Figure 3.2: Sketches for designing the building. (a) Sketch-1. (b) Sketch-2. (c) Sketch-3.

Figure 3.3: Work flow of designing a building. (a) Step-1. (b) Step-2. (c) Step-3. (d) Step-4.
Chapter 4

Overview of System

Our overall design philosophy focuses on simplicity, e.g., in terms of interface design (so as to reduce mental load of the user) and geometric modeling paradigm, as well as close conformation to the traditional design habits of architects. The whole design work flow, allowing a user to switch back and force, is a simulation of traditional architectural design progress, plus letting users perform 3D operations, e.g., 3D editing or observation, etc., in an interactive design environment. The system is coded using VC++ 2005 with OpenGL and the graphical user interface is programmed under the latest GLUI.

4.1 Supported model creation

For 3D models, we can roughly separate them into three categories, i.e., Extrusion, Primitive and Free-form models. A structured classification can be found in Figure 4.1. More detailed description can be found in Chapter 5 and Chapter 6.

The most intuitive way to create 3D model parts of an architectural entity is the Extrusion method, for it totally conforms to traditional architectural design habits using papers and pencils. It is a pure sketch-based operation function set. Users can simply sketch a 2D close contour, which is a polygon or has free curves embedded in it, and then make it an geometric entity by giving it a height. Using this Extrusion method to create architectural models, users can experience the model-growing progress conforming to architects' design habit and thinking speed.

Another useful way to supplement the Extrusion operation is Primitive method. When the objective models, or parts of them, are absolutely clear for the designers, e.g., spheres or
pyramids, etc., the most efficient way is to directly draw them out with simple operations on the screen. For the reason that primitive geometric entities are widely used in architectural design, this Primitive method is as important as the Extrusion one. Compared with standard industrial modeling software, such as 3ds MAX and AutoCAD, our Primitive method allows free model transformation in some degree, e.g., from a cylinder to a cone, or from a prism to a frustum pyramid.

In architectural design, we seldom use Free-form model to construct architectural entities, so this category of model is excluded out from our modeling system.
4.2 Hardware Interface

Typically, in CAGD software, such as AutoCAD and 3ds MAX, etc., users utilize the keyboard and mouse to interact with computers. This is quite useful on detailed input and accurate pointing. But in the stage of conceptual design, these hardware devices become the barrier between users and computers in some degree. We can often be put into an awkward situation when we try to draw free lines on screen with a mouse, and frequently moving the hand from mouse to keyboard for inputting some precise numbers to construct a model is also a cumbersome experience.

In our system, we try to utilize the hardware features of Tablet PC to design our hardware interactive interface. Our final goal is to totally abandon the keyboard and mouse, and use a single Tablet PC stylus (the digital pen) to complete all tasks instead. Currently, keyboard has been excluded out from our system. With the purpose of general testing, we still use the mouse to simulate the operation of a stylus, thus makes it possible to develop and test our system with a common PC. At the same time, we always keep the stylus operational style in our mind, and each function utility we define on the mouse can be mapped to a two-key stylus. Table 4.1 shows detailed operation mapping between the mouse and stylus.

- Press-down-move mouse operation:

Besides standard mouse operation, such as mouse click and button press down, the press-down-move operation is a basic mouse operation used in our system. Almost all the modeling functions, either for object creation or for object modification, are related to the press-down-move mouse operation. This press-down-move operation is done by pressing down a mouse button and then move the mouse with that button still pressed down. This is a typical mouse operation used in our system and we will frequently mention it in the following sections and chapters.

4.3 Mouse Operations

With the abandonment of the keyboard, we use the mouse to simulate the operations of the Tablet PC stylus. Currently, all the mouse operations needed in our prototype system are presented in Table 4.1.
Table 4.1: Function key mapping between a mouse and a Tablet PC stylus. Only these simple stylus functions listed in this table are needed to fulfill the whole operation in our current system. This shows the high efficiency of using the stylus. On the other hand, We can notice that only a very limited function combination of the Tablet PC stylus is used in our current system. That means the function of the stylus has a great potential to be extended when new operation features are added.

Different mouse operations in different program modes correspond to different functions. We try to devise a set of direct, intuitive and natural mouse/stylus operations to conform to human’s drawing behavior mode and architects’ traditional design habits. With the basic program design philosophy of using the features of a two key Tablet PC stylus, and the mapping relationship between the mouse and the stylus showed in Table 4.1, we arrange all the mouse operations needed in our system in Table 4.2.

4.4 Software Interface

The system interface is showed in Figure 4.2. Left part is the main viewport of the 2D/3D workspace. All interactive operations between the user and computer are done in this viewport. Right part shows the control panels for setting the work environment. In Primitive panel, users can select Column or Ellipsoid option to create desired 3D primitive entities. If the Primitive function is not activated, the default creation mode is set for creating 3D Extrusion models.

Display panel is used for displaying models in various modes. Users can make the system display object frames or triangles. Opaque, Translucent, Transparent and Hide-line effects can be displayed. Shading effect can be designated by users for better observation. Reference marks can also be displayed according to the desire of the users. Setting panel provides a
CHAPTER 4. OVERVIEW OF SYSTEM

<table>
<thead>
<tr>
<th>Actions</th>
<th>Mouse Operations</th>
<th>Workspace and Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>switch 2D/3D mode</td>
<td>right button click</td>
<td>2D/3D</td>
</tr>
<tr>
<td>zoom in/out</td>
<td>right button press-down-move</td>
<td>2D/3D</td>
</tr>
<tr>
<td>pan</td>
<td>middle button press-down-move</td>
<td>2D/3D</td>
</tr>
<tr>
<td>switch Sketch/Draw</td>
<td>left button click on blank space</td>
<td>2D, no element selected</td>
</tr>
<tr>
<td>select/deselect elements</td>
<td>left button click on elements</td>
<td>2D</td>
</tr>
<tr>
<td>deselect all elements</td>
<td>left button click on blank space</td>
<td>2D, elements selected</td>
</tr>
<tr>
<td>draw sketches/lines/curves/primitives</td>
<td>left button press-down-move</td>
<td>2D, no element selected</td>
</tr>
<tr>
<td>move control points</td>
<td>left button press-down-move on control points</td>
<td>2D, elements selected</td>
</tr>
<tr>
<td>move elements</td>
<td>left button press-down-move not on control points</td>
<td>2D, elements selected</td>
</tr>
<tr>
<td>change extrusion model heights</td>
<td>left button press-down-move</td>
<td>3D, extrusion model top surfaces selected</td>
</tr>
<tr>
<td>change extrusion model elevations</td>
<td>left button press-down-move</td>
<td>3D, extrusion models selected</td>
</tr>
<tr>
<td>transform primitive models</td>
<td>left button press-down-move</td>
<td>3D, primitive model control points selected</td>
</tr>
<tr>
<td>change primitive model elevations</td>
<td>left button press-down-move</td>
<td>3D, primitive models selected</td>
</tr>
</tbody>
</table>

Table 4.2: Mouse Operations used in our modeling system. Different mouse operations in different program modes correspond to different functions.

The lower-right part of the control panels are Import/Export modules. For a modeling system towards practical use, the Import/Export feature is very important. We designed a file format, 'sbm', for users to work in our system, so we can output current workspace information, as much as possible, to this 'sbm' file. This file allows users load in all the information needed, i.e. geometrical values and environmental parameters, etc., into our system, and then users can continue their previous work from the exactly same status where they left the program last time. Another export file format is the Wavefront 'obj'. Using this file format, we can load in the models created in our system into 3ds MAX. This makes our system connected with some popular commercial application and we think this
is a good extension of our system.

4.5 Basic Modeling Functions

In Extrusion mode, users can draw free sketches on an infinite virtual drawing canvas. System can recognize line segment, curves, and can automatically connect them into close polygons, curved contours, or composite contours. 3D models can be created by simply giving contours heights. Both 2D elements and 3D objects can be modified freely. Detailed description can be found in Chapter 5.
After activate the *Primitive* mode, two general classes of 3D primitive entities can be created. One is *Column*, another one is *Ellipsoid*. In *Column* mode, models with polygonal section, such as *Prism*, *Pyramid* and *Frustum Pyramid* can be created with desired edge numbers. Objects created with polygonal section can be converted mutually. Models with ellipse section, such as *Cylinder*, *Cone* and *Frustum Cone* can also be created in the *Column* mode with wanted precision. Models created with ellipse section can also be converted mutually. In *Ellipsoid* mode, ellipsoids with needed smoothness can be created. Free modification of the ellipsoid is supported in our system. Detailed description can be found in Chapter 6.
Chapter 5

Extrusion Method

In our prototype system, we implemented some basic features of behaviorized modeling software based on sketches. Extrusion Method is the basic operation used in our sketch-based modeling system. Like architects using tracing papers and pencils to do conceptual design, our system provides users an infinite digital canvas and lets designers to draw various sketches on it as freely as they work on real papers.

Our prototype system can capture, record, and display sketches, perform line segment and cubic Bezier curve construction from sketches, automatically connect lines to poly-lines, construct closed poly-lines to base polygons, and finally, extrude a base polygon to form a 3D object. As the parts of a whole complex model, these 3D objects can be used to assemble together, and finally be constructed into an integrated 3D object we want.

5.1 Creation operation in Extrusion module

5.1.1 Sketch capture, recording and displaying

When we press-down-move the left mouse button under the sketch mode, discrete cursor track points on the screen are captured. The Screen Coordinates System (SCS) values are converted into World Coordinates System (WCS) and are recorded. The corresponding continuous sketch is displayed by drawing line segments between adjacent captured points. Sketches are many sets of line strips connected by captured screen points. We call a continuous sketch a stroke. Figure 5.1 shows a stroke with red points illustrating captured screen points.
Figure 5.1: A stroke is constructed by a continuous sketch. Strokes are used to recognize our sketches to line segments or curves. Red points on the sketch illustrate captured screen points.

5.1.2 Line segment recognition

Figure 5.2: (a) Line segment recognition. Blue rectangles illustrate the test envelopes given by a user-defined threshold (thickness of the envelope). The threshold can be automatically adjusted based on stroke length. If all captured points along the stroke fall inside the corresponding envelope, a line segment is recognized. (b) Recognition of axis-aligned line segments. When we want to draw an axis-aligned line segment, it is hard to keep the stroke drawn absolutely horizontal or vertical. With acceptable tolerance, a line close to being axis-aligned will be converted to one automatically.

When a stroke is captured, it is first tested whether it should be treated as a (straight) line segment. A virtual line segment is first formed by connecting the start and end points of the current stroke. Distances between each captured point along the stroke and the virtual line segment are recorded and tested against a user-defined threshold. If all distances are within the threshold, the stroke is regarded as a line segment which is given by the already constructed virtual line segment. Figure 5.2 (a) shows several recognized line segments.

The threshold value can be automatically adjusted based on the length of the stroke so
that a reasonable hand shaking can be tolerated. The threshold value $\varepsilon$ is given by

$$\varepsilon = \frac{D_{\text{line}}}{D_{\text{screen}}} \cdot R_{\text{long}} + R_{\text{short}}$$  \hfill (5.1)

where $D_{\text{line}}$ is the pixel-distance of current virtual line segment, $D_{\text{screen}}$ is diagonal pixel-distance of the screen viewport, $R_{\text{long}}$ is the long recognition parameter, and $R_{\text{short}}$ is the short recognition parameter. Experimentally, setting $R_{\text{long}}$ to 40 and $R_{\text{short}}$ to 5 can bring us a satisfied result.

In addition, when a recognized line segment makes a sufficiently small angle with either the (world) horizontal or vertical axis, it will automatically be treated as a horizontal or vertical line segment, respectively. The reason for performing such a line transformation is that in architectural design, lots of axis-aligned line segments will be drawn. However, sketching out precisely axis-aligned lines is not easy with either a mouse pointer or a stylus. This functionality allows the construction of horizontal or vertical line segments with acceptable tolerance. Figure 5.2 (b) shows a few recognized axis-aligned line segments.

### 5.1.3 Cubic Beziér curve construction from sketches

![Figure 5.3: (a) Single piece cubic Beziér curve. Control points $p_1$, $p_2$, $p_3$ and $p_4$ determine this Beziér curve. In order to manipulate this curve, we should find all these four control points. (b) Single piece cubic Beziér curve subdivision. We use a binary tree data structure to store control points. Data on the top of leaves are what we need to draw cubic Beziér curves. Data on the root are control points used to manipulate cubic Beziér curves.](image)

If a stroke can not be recognized as a line segment, it will be approximated as a single
piece cubic Bezier curve. Typically, most curves we draw in practical designs are single piece cubic Bezier curves, because we do not need too many bendings in a curve. Cubic Bezier curve is a well studied polynomial curve. As showed in Figure 5.3 (a), a cubic Bezier curve is defined by four control points, p_1, p_2, p_3 and p_4. Once four control points are made certain, a cubic Bezier curve is uniquely defined. Given four control points, a cubic Bezier curve can be described as an equation:

\[
B(t) = p_1 \cdot (1 - t)^3 + p_2 \cdot [3t(1 - t)^2] + p_3 \cdot [3t^2(1 - t)] + p_4 \cdot t^3 \quad (0 \leq t \leq 1) \tag{5.2}
\]

Obviously, when we approximate a stroke as a single piece cubic Bezier curve, the start and end points of our stroke are known as p_1 and p_4. So, we have two unknowns, p_2 and p_3, which are showed in boldface in the following equations. Suppose we have other two points, q_2=B(t_2) and q_3=B(t_3) as showed in Figure 5.3 (a), based on equation 5.2, we can get:

\[
q_2 = B(t_2) = p_1 \cdot (1 - t_2)^3 + p_2 \cdot [3t_2(1 - t_2)^2] + p_3 \cdot [3t_2^2(1 - t_2)] + p_4 \cdot t_2^3 \quad (0 \leq t_2 \leq 1) \tag{5.3}
\]

\[
q_3 = B(t_3) = p_1 \cdot (1 - t_3)^3 + p_2 \cdot [3t_3(1 - t_3)^2] + p_3 \cdot [3t_3^2(1 - t_3)] + p_4 \cdot t_3^3 \quad (0 \leq t_3 \leq 1) \tag{5.4}
\]

Let us define coefficients of these four control points, p_1, p_2, p_3, p_4, as:

\[
c_{1.q2} = (1-t_2)^3; \quad c_{2.q2} = 3 \cdot t_2 \cdot (1-t_2)^2; \quad c_{3.q2} = 3 \cdot t_2^2 \cdot (1-t_2); \quad c_{4.q2} = t_2^3.
\]

\[
c_{1.q3} = (1-t_3)^3; \quad c_{2.q3} = 3 \cdot t_3 \cdot (1-t_3)^2; \quad c_{3.q3} = 3 \cdot t_3^2 \cdot (1-t_3); \quad c_{4.q3} = t_3^3.
\]

Then, we can write equation 5.3 and 5.4 as:

\[
q_2 = B(t_2) = p_1 \cdot c_{1.q2} + p_2 \cdot c_{2.q2} + p_3 \cdot c_{3.q2} + p_4 \cdot c_{4.q2} \quad (0 < t_2 < 1) \tag{5.5}
\]

\[
q_3 = B(t_3) = p_1 \cdot c_{1.q3} + p_2 \cdot c_{2.q3} + p_3 \cdot c_{3.q3} + p_4 \cdot c_{4.q3} \quad (0 < t_2 < 1) \tag{5.6}
\]

So, we have two unknowns, p_2 and p_3, in two equations 5.5 and 5.6. By solving these two equations, we can get:

\[
p_2 = \frac{[q_2 \cdot c_{3.q3} - q_3 \cdot c_{3.q2} - p_1 \cdot (c_{1.q2} \cdot c_{3.q3} - c_{1.q3} \cdot c_{3.q2}) - p_4 \cdot (c_{4.q2} \cdot c_{3.q3} - c_{4.q3} \cdot c_{3.q2})]}{(c_{2.q2} \cdot c_{3.q3} - c_{2.q3} \cdot c_{3.q2})} \tag{5.7}
\]

\[
p_3 = \frac{[q_2 \cdot c_{2.q3} - q_3 \cdot c_{2.q2} - p_1 \cdot (c_{1.q2} \cdot c_{2.q3} - c_{1.q3} \cdot c_{2.q2}) - p_4 \cdot (c_{4.q2} \cdot c_{2.q3} - c_{4.q3} \cdot c_{2.q2})]}{(c_{3.q2} \cdot c_{2.q3} - c_{3.q3} \cdot c_{2.q2})} \tag{5.8}
\]
CHAPTER 5. EXTRUSION METHOD

If we can find \( q_2 \) and \( q_3 \) on a cubic Beziér curve and get their corresponding \( t_2 \) and \( t_3 \), all the coefficients can be solved, and then we can solve equation 5.7 and 5.8. Unfortunately, given a point \( B(t) \), finding its corresponding \( t \) without adequate information of all four control points is non-trivial. In general, the relationship between \( B(t) \) and \( t \) will be non-obvious, with \( t \) changing faster along the “more bent” portions of the curve.

So, an approximation is devised in our prototype system to find \( B(t) \) and its corresponding \( t \). Let us define \( l(B([a, b])) \) to be the length of the Bezier curve over the interval \([a, b]\), where \([a, b]\) is a subinterval of \([0, 1]\). The basic idea of this approximation can be described as:

\[
t = \frac{l(B([0, t]))}{l(B([0, 1]))}
\]

(5.9)

where \( l(B([0, t])) \) is the length between \( B(0) \) and \( B(t) \) along the curve, \( l(B([0, 1])) \) is the total length of a cubic Bezier curve.

In our program, we first calculate the total length \( L \):

\[
L = \text{summation of all distances between two immediately adjacent points from } P_1 \text{ to } P_4
\]

Then, we calculate two lengths \( L_2 \) and \( L_3 \):

\[
L_2 = \text{summation of all distances between two immediately adjacent points from } P_1 \text{ to } q_2
\]

\[
L_3 = \text{summation of all distances between two immediately adjacent points from } P_1 \text{ to } q_3
\]

So, based on equation 5.9, we have:

\[
t_2 = \frac{L_2}{L}
\]

(5.10)

\[
t_3 = \frac{L_3}{L}
\]

(5.11)

For a general approximation, we try to find \( q_2 \) and \( q_3 \) such that \( t_2 \) gets close to 1/3 and \( t_3 \) gets close to 2/3. After finding \( q_2 \) and \( q_3 \) and their corresponding \( t_2 \) and \( t_3 \), equations 5.7 and 5.8 can be solved and finally we can get all four control points, \( P_1, P_2, P_3 \) and \( P_4 \).

Once the four control points of the cubic Bezier curve are computed, we render the curve via subdivision according to the well known deCastlaju algorithm [6]. As showed in Figure 5.3 (b), after one level of subdivision, we get two sets of refined control points. One is \( \{r_1, r_2, r_3, r_4\} \) and another is \( \{r_5, r_6, r_7, r_8\} \). And we can get a subdivision point \( r_4 = r_5 = B(1/2) \).

Connect \( r_1, r_4 (r_5) \) and \( r_8 \), we get a one level subdivided cubic Bezier curve. Continue doing subdivision with the new control points to a required level, we can get a set of control points
and subdivision points at different levels. Sequentially connect these subdivision points, we can approximate a single piece cubic Beziér curve with a line strip.

5.1.4 Poly-line construction

We refer to a line strip as a poly-line. A closed poly-line is a base polygon which can be used to create a 3D object by giving a height, the thickness of the object, to it; this is the extrusion process. A new line segment should be automatically joined to a poly-line if its end points are sufficiently close to other open end points present in the workspace. We call this operation line segment end point intersection detection and the threshold used for doing this detection can be set by the users. When a new line segment is added into the workspace, one or two end point intersections may be found by it. In Figure 5.4, new line segments are shown in red covered with small blue blocks, while open poly-lines are shown with green color and closed poly-lines are shown in blue.

![Figure 5.4: A new line segment finds one or two end intersections.](image)

(a) A line segment intersects with another line segment. (b) A line segment intersects with a poly-line. (c) A line segment intersects with other two line segments. (d) A line segment intersects with a line segments and a poly-line. (e) A line segment intersects with two poly-lines. (f) A line segment intersects with a poly-line and close it.

5.1.5 Polygon triangulation and extrusion

When a poly-line is closed, it turns to be a base polygon. Polygons in 2D are tessellated by the well known Delaunay triangulation. The general 3D Delaunay criterion, sometimes called the "empty sphere" property, says that any node must not be contained within the circumsphere of any tetrahedron within the mesh. A circumsphere can be defined as the sphere passing through all four vertices of a tetrahedron. To triangulate a 2D polygon, we
only need to deal with 2D vertices. Figure 5.5 (a) and (b) is a simple 2D illustration of Delaunay criterion. (c) and (d) are two triangulated polygons computed by our system.

![Figure 5.5: 2D polygon triangulation. (a) maintains the Delaunay triangulation criterion while (b) does not. (c) and (d) are two triangulated polygons computed by our system.](image)

Figure 5.6: Extruding 2D polygons to 3D objects. (a) and (b) show 2 models created by the Extrusion operation.

![Figure 5.6: Extruding 2D polygons to 3D objects. (a) and (b) show 2 models created by the Extrusion operation.](image)

After polygon triangulation, we can finally extrude a closed 2D polygon into a 3D object by specifying a height for it. This Extrusion operation is done in the 3D workspace. Users can press-down-move left mouse button in the 3D Observation Mode. The vertical movement of the mouse can be captured and the move distance can be computed. The move distance in the vertical direction is then passed to the selected closed 2D polygon to do the Extrusion operation. This sketch-based Extrusion method is the basic way to create 3D models using our prototype system. We can look those extrusion models as the parts of a complex 3D entity and achieve the objective 3D entity by assembling these parts. Figure 5.6 shows two sample models created by the Extrusion method.
5.2 Modification operation in Extrusion module

5.2.1 Colors used on the Extrusion elements

Line Segment, Open Poly-line, Closed Polygon and 3D Extruded Entity, are basic Extrusion elements used in our system for creating and modifying 3D geometric models. Because we provide different operation function to different elements, it is important for users to clearly identify them without ambiguity and confusion. To achieve this goal, we use different colors to indicate different elements and that makes it easy for users to manipulate them. Figure 5.7 illustrates different colors used in our system for different geometric elements.

![Different colors used on different geometric elements](image)

Figure 5.7: Different colors used on different geometric elements. (a) Line Segment. (b) Open Poly-line. (c) Closed Polygon. (d) Unselected 3D Extruded Entity. (e) Surface selected 3D Extruded Entity. (f) Body selected 3D Extruded Entity.

5.2.2 Extrusion element Selection/Deselection

![Extrusion element selection in 2D workspace](image)

Figure 5.8: Extrusion element selection in 2D workspace. (a) Line Segment. (b) Open Poly-line. (c) Closed Polygon or 3D extrusion entity.

If a user wants to modify an Extrusion element, the element should be selected first. In this prototype system, we try to use a way to accomplish element Selection/Deselection.
function as easily and intuitively as possible. A user can select Line Segment, Open Poly-line and 3D Extruded Entity by a simple click on any part of them. To select the top surface of an 3D Extruded Entity, the user only need to click on that surface. Multiple selection with the same element type is supported by our prototype system.

After elements are selected, they turn to be displayed with dashed lines to separate them from other elements. All control points on those selected elements, such as end points of line segments and control points of cubic Beziér curves, are displayed as small handles (red empty squares) for users to manipulate the elements. Control points of cubic Beziér curves are connected with fine dashed control lines to indicate the curve scopes. These fine dashed control lines are sequentially grouped with red and blue colors to separate immediately adjacent cubic Beziér curves apart to avoid confusion brought by line overlapping. Figure 5.8 shows these control points and lines of different elements after they are selected in 2D workspace.

To deselect a single selected element, a user only need to click on it again with the same operation used to select it. To deselect all the selected elements in the workspace, the user can do a simple click on any blank part of the virtual design canvas.

5.2.3 Extrusion element and control point Movement in 2D workspace

![Extrusion element and control point Movement in 2D workspace](image)

Figure 5.9: Extrusion element and control point Movement in 2D workspace. (a)-(c) Move control points of Line Segment, Open Poly-line and Closed Polygon or 3D extrusion entity. The moved control points are showed as solid red squares. (d)-(f) Move Line Segment, Open Poly-line and Closed Polygon or 3D extrusion entity.

If a user wants to move elements in 2D workspace, a press-down-move operation is
CHAPTER 5. EXTRUSION METHOD

needed. To move an end point of a selected line segment, or move a control point of a selected cubic Bezier curve, a user can move the mouse cursor on a wanted control handle (the red empty square), and then press-down-move the left mouse button. During the period of movement operation, the manipulated line segment or cubic Bezier curve is dynamically updated corresponding to the movement at an interactive rate. After the pressed control point is moved to a desired position, the user can release the pressed mouse button to terminate the movement. Figure 5.9 (a)-(c) illustrates some control point movement operations in 2D workspace.

To move selected elements in 2D workspace, no matter what type they are of, such as Line Segment, Open Poly-line, Closed Polygon and 3D Extruded Entity, a user can move the mouse cursor to a blank position on the workspace and do left button press-down-move operation. Multiple selected element movement is supported in our system. During the period of movement operation, the whole selected elements are all dynamically updated to new positions according to the movement in real time. After those elements are moved to the objective place, the user can stop this movement operation by releasing the pressed left button. Figure 5.9 (d)-(f) shows some element movement operation in 2D workspace.

5.2.4 Extrusion element Connection

![Extrusion element Connection](image)

Figure 5.10: Extrusion element Connection. (a) A line segment intersects with a line segment. (b) A line segment intersects with a poly-line. (c) A line segment intersects with two line segments. (d) A line segment intersects with a line segments and a poly-line. (e) A poly-line intersects with another poly-line. (f) Self-close a poly-line.
CHAPTER 5. EXTRUSION METHOD

After element or control point movement finishes, end point intersection detection will occur, using the same criterion presented in Section 5.1.4. Any open end point, no matter on line segments or open poly-lines, can connect to any other open end point if these two end points are not on the same line segment. Using this way, we can connect elements by movement just like we draw new elements into the workspace. Figure 5.10 illustrates some connection results after Extrusion element or control point movement occurs in 2D workspace.

5.2.5 *Extrusion* surface and object Movement in 3D workspace

![Extrusion surface and object Movement in 3D workspace](image)

Figure 5.11: *Extrusion* surface and object Movement in 3D workspace. (a) Move the top surface to change the height/thickness of an Extrusion object. (b) Move an entire Extrusion object in the space to change its elevation.

While the 2D Extrusion elements, such as line segments and cubic Beziér curves, can be modified in a variety of ways, as illustrated in previous sections, our system also allows for modifications in 3D workspace.

For example, we can change the height/thickness of an object by moving its selected top surface. After select the top surface of an object in the 2D workspace, we can switch to the 3D workspace, the *Observation Mode*, with a right mouse button click. In the 3D workspace, a user can press-down-move left mouse button to dynamically change the height of this selected object. The vertical movement of the mouse can be captured and the move distance can be computed, just like what we do for extruding a closed polygon to an object in Section 5.1.5. The move distance in the vertical direction is then used to adjust the space position of the selected surface. This operation can be terminated after the object height is right adjusted by releasing the pressed left mouse button. This height movement operation is depicted in Figure 5.11 (a).
To change the elevation of an extrusion object, e.g., to move an object from the floor up into space vertically, we can simply select the entire object and move it in 3D workspace. The elevation movement operation of an extrusion object is similar to the height movement. An example of this elevation movement operation is showed in Figure 5.11 (b).

5.3 Modeling aspects in Extrusion module

5.3.1 Sketch-retained design process

Figure 5.12: Sketch-retained design process. The Fallingwater Villa model (a) is displayed in translucent mode together with the sketches on the floor plan. An imaginary architectural model and the design sketches showed in (b).

Sketches represent our design thoughts. It plays a very important role in conceptual designs. Architects draw a lot of sketches during the process of doing the conceptual designs, and finally extract useful information from them. Sketches can record the swift ideas welling up in architects’ minds and provide thinking hard copies for designers to develop, judge, modify and compare their designs. Because the design process is time dependent, an architect often goes back to his/her original sketches to retrieve valuable design ideas for the current use. The piles of tracing papers full of various sketches on an architect’s design table is a good evidence.

Our modeling prototype system simulates this practical design method. Sketches drawn on the virtual design canvas are retained in the workspace. These sketches can be displayed together with the 3D models created based on them, and provide users very useful references
for the design comparison and judgement. A check option is provided in our system to switch on/off the sketch display in the workspace, just like an architect puts the sketch tracing papers under his/her final design drawings or pulls them out. Figure 5.12 depicts this function with two examples.

5.3.2 Cubic Beziér curve modification

![Cubic Beziér curve modification](image)

Figure 5.13: Cubic Beziér curve modification. In this figure, two examples showed in (a) and (b) illustrate the cubic Beziér curve modification operation.

In most cases, our cubic Beziér curve fitting algorithm works well and can satisfy practical application. But, because this algorithm is an approximate solution, sometimes the recognized curve does not fit the curve stroke quite well and needs a modification operation to refine it. In our prototype, it is a easy job to fulfill this requirement. A user only need to press-down-move the curve control points and the system will respond a dynamically updated feedback at an interactive rate. Two curve modification examples are showed in Figure 5.13.

5.3.3 Curve smoothness

In our prototype system, a curve sketch stroke can be recognized as a single piece cubic Beziér curve. In practice, 3 subdivision levels or beyond are typically sufficient for our curve approximation tasks. Our system provides three smoothness options for a cubic Beziér curve, corresponding to the subdivision level 3, 4 and 5. Figure 5.14 illustrates an extrusion object with a piece of curved side surface created using 3 smoothness options.
5.3.4 Flexible feature tolerance at curve joints

When two cubic Bezier curve pieces join, we sometimes wish to preserve a sharp feature. At other times, we may want the curves to achieve $C^1$ or $G^1$ continuity. These requirements are both easy to meet in our prototype system by properly manipulating the Bezier curve control points. We can find the sharp feature is kept at the joint of two Bezier curves in Figure 5.15 (a). Another example showed in Figure 5.15 (b) depicts how to make the joint keep the $C^1$ or $G^1$ continuity.

5.3.5 2D/3D dynamic synchronization

If we change the base polygon shapes in 2D workspace, corresponding 3D extrusion objects will be changed simultaneously. Both the bottom and top surfaces will be changed according to the 2D modification. The height/thickness of the extrusion object will retain. Figure 5.16 shows this 2D/3D synchronization function.
CHAPTER 5. EXTRUSION METHOD

Figure 5.15: **Flexible feature tolerance at curve joints.** The sharp feature is kept at the joint of two Bezier curves in (a), and (b) depicts how to make the joint keep the $C^1$ or $G^1$ continuity.

Figure 5.16: **Flexible feature tolerance at curve joints.** An initial extrusion object is showed in (a), while the modified 2D and 3D views showed in (b) after changing the base polygon shape in 2D workspace.
Chapter 6

Primitive Method

Architects often use primitive models, such as spheres, pyramids and cylinders, etc., as parts to construct complex building models, together with the extrusion method. We can easily find some primitive models or their transmutations, e.g., round corridor pillars and sphere-like auditoriums, etc., existing in real architectures.

In most circumstance, using extrusion method or other ways, instead of utilizing a special primitive object creation method, to create primitive objects is not an efficient or feasible solution. Sometimes, those methods bring ambiguity into the primitive model creation process, and sometimes, those methods are not competent enough for some specific primitive object creation tasks. Based on this analysis, in our prototype modeling system, a special Primitive model creation module is provided as the primitive model creation solution.

A good feature of the primitive models is that they can be efficiently expressed with mathematical parameters. These parameters are quite useful on primitive model creation and modification processing. For example, an elliptical section and a height value are sufficient enough for us to create a cylinder. Another correspondingly non-trivial instance is, we can only use one center point and six axes to represent an arbitrary ellipsoid. In our prototype, we use various suitable parameters to control all the primitive models supported in the system, and make them transparent to users based on the behaviorized software design philosophy. When users manipulate primitive models in our modeling environment, they actually control those primitive model parameters embedded beneath the user interface, although they don’t need to input any parameter in the whole modeling process. What they need to interact with the system to create and modify the wanted primitive models are only the model control points/frames.
6.1 Primitive model classification

Currently, some column-like and ellipsoid-like primitive objects can be created in our prototype system. A list of supported primitive objects can be found in Figure 4.1.

We divide column-like primitive objects into two parts based on the types of their sections, one is polygon section, and another is ellipse section. Essentially, these two types of sections play the same role for creating column-like models. Both of them are used to control the section edge numbers or smoothness. The purpose of providing this division is to make it easy for users to control the section. For the polygon section, users can specify the value of section edge number from 3 to 15. For the ellipse section, users can choose one of the three smoothness precision, which are coarse, moderate and fine, corresponding to the section edge number value 16, 32 and 64. In the polygon section option, primitive objects *Prism*, *Pyramid* and *Frustum Pyramid*, etc., can be created. Figure 6.1 (a)-(c) shows some examples of these polygon section column-like objects. In the ellipse section option, other primitive objects with more smooth sections, such as *Cylinder*, *Cone* and *Frustum Cone*, etc., can be modeled. Figure 6.1 (d)-(f) illustrates some examples of these ellipse section column-like objects.

Figure 6.1: Column-like primitive objects. (a) *Prism*, section edge number: 4. (b) *Pyramid*, section edge number: 8. (c) *Frustum Pyramid*, section edge number: 7. (d) *Cylinder*, smoothness precision: coarse (section edge number:16). (e) *Cone*, smoothness precision: moderate (section edge number:32). (f) *Frustum Cone*, smoothness precision: fine (section edge number:64).

Ellipsoid-like primitive objects are also supported by our prototype system. The six half-axes of an ellipsoid-like object can be of different lengths, but they are required to converge to a same point, the center of the ellipsoid-like object. By adjusting the lengths of these ellipsoid half-axes, users can obtain different ellipsoid-like primitive objects. All these objects conform to those same requirements described above, although they can look quite
CHAPTER 6. PRIMITIVE METHOD

different from each other. Figure 6.2 (a)-(f) depicts some examples of the ellipsoid-like primitive objects.

Figure 6.2: Ellipsoid-like primitive objects. (a)-(f) Objects created using different half-axes. All models are of the moderate smoothness precision. The six half-axes of an ellipsoid-like object can be of different lengths, but they are required to converge to a same point, the center of the ellipsoid-like object. All these objects conform to same requirements, although they may look quite different from each other.

6.2 Common properties about Primitive models

6.2.1 Control frame used on Primitive models

Figure 6.3: Primitive object control frames. (a) A Prism with control frame. (b) A Frustum Cone with control frame. (c) An Ellipsoid with control frame. (d) The control frame of object (c).

Control frames are used to control the shapes of Primitive models. A Control frame consists of control points, control lines, control ellipses or ellipse pieces. The control points are used for users to change the object shapes as handles. The control lines are used to connect control points and indicate the geometric features of a primitive model, such as the control ellipse axes, the axis lengths and the relationship between these axes. The control ellipses are used to indicate the wraps, which are the circum-ellipses or inscribed-ellipses of
the top and bottom surfaces of a column-like primitive object. The control ellipse pieces are used to determine the shape of an ellipsoid-like model. Figure 6.3 shows some primitive models and their control frames.

6.2.2 Colors and line types used on control frames

![Figure 6.4: Colors used on primitive object control frames. (a) A Frustum Pyramid with control frame. (c) An Ellipsoid with control frame. (d) The control frame of object (b).](image)

Different colors and line types are used to indicate different status of the control frames. Users can get needed object geometric information based on these colors and line types. Figure 6.4 illustrates some examples.

In (a), a Frustum Pyramid with control frame are showed to indicate the colors and line types used for column-like objects. Control points on the top surface are showed with empty blue squares, and solid red squares are used to represent control points of the bottom surface. If two axes of a control ellipse are of the same lengths, both of these two axes are displayed with green bold dashed lines and we can find an instance on the top surface. Otherwise, the two axes, which are of different lengths, of a control ellipse are displayed with fine red and blue dashed lines. Users can easily adjust these surfaces to desired shapes by dynamically observing the status of these axes when they move the control points on the frame.

In (b) and (c), an Ellipsoid and its control frame are showed to indicate the colors and line types used for ellipsoid-like objects. All control points are displayed as solid red squares. Equal length axes are showed using bold solid lines and different colors are used to group axes with equal lengths. Axes showed with fine dashed lines indicate there is no any other axis which is of the same length. By observing these axes along with the movement of
control points, a user can easily change an arbitrary ellipsoid back into a regular sphere or create a partly symmetric ellipsoid.

6.2.3 Control point auto snap features

![Control point auto snap features](image)

Figure 6.5: Control point auto snap features. (a)-(b) Column object control point auto snap. (c)-(d) Ellipsoid object control point auto snap.

When we modify a primitive model, we often want a control point on the control frame to be accurately put to the exact position of another control point on the same control frame in 2D or 2.5D (2D projection of control points with different height) domain. This can be easily done by an Auto-Snap mechanism provided by our system. The Auto-Snap mechanism can force a being-moved control point snapping to another control point on the same control frame if they are close enough. Figure 6.5 depicts two simple application of this auto-snap function.

An example of column-like object auto-snap is showed in Figure 6.5 (a)-(b). When a user moves the lower-right control point of the top surface, it is “automatically snapped” to the lower-right control point of the bottom surface when the projection of these two control points get close enough to each other, and thus makes the object change from a Frustum Pyramid (Figure 6.5 (a)) to a Prism (Figure 6.5 (b)).

Another example showed in Figure 6.5 (c)-(d) illustrates the auto-snap function applied on ellipsoid-like objects. In this case, the lower-right control point of the horizontal control ellipse pieces is “automatically snapped” to the right middle control point of the horizontal
control ellipse pieces when these two control points get close enough to each other, and thus makes the object change from a round ellipsoid (Figure 6.5 (c)) to a clipped one (Figure 6.5 (d)).

6.2.4 Model precision adjustment

We can directly change the precision of a selected primitive object by simply changing the creation option in the corresponding menu, without changing its control frame. We at least have two reasons to provide this function. One reason is for Storage Efficiency. For example, we only need to create a coarse model when it is a small part of a big object. So, we need to change some parts of a complex model from Fine/Moderate precision to Moderate/Coarse precision. Another reason is for Time Efficiency. Sometimes, although we want to create a fine precision model as a part of a big object, we don’t need to set it to be of fine precision during the creation period, because creating a fine model is more time-consuming than creating a coarse one. Instead of doing so, we can create and modify a coarse model in the design process and finally change it to be a fine precision object. Figure 6.6 illustrates some examples of this precision changing application.

6.3 Creation operation in Primitive module

Primitive model creation is a very simple operation in our system. After activating appropriate creation options in the menu, a user can easily create a Primitive model by a single left
mouse button press-down-move operation. During the creation process, a 2D basic control frame dynamically updates according to the mouse movement, and the user can adjust this control frame to a wanted status based on the geometric information of the control ellipse and its axes in the control frame. After releasing the mouse button, a Primitive object is created. Figure 6.7 shows some examples of this Primitive model creation processing. In this figure, a 3-edge Prism with regular triangle section, a coarse precision Frustum Cone with ellipse section, and a partly symmetric Ellipsoid with moderate precision are created. The 2D and 3D shapes are depicted in the upper and lower rows respectively.

A Primitive Ellipsoid is constructed by stitching eight Ellipsoid Pieces together. Each ellipsoid piece is determined by three half-axes. At the intersections of these eight ellipsoid pieces, $C^1$ or $C^1$ continuity is maintained to keep the ellipsoid smooth when we want to model a convex ellipsoid. Because the three half-axes of an ellipsoid piece are independent to their neighbor axes, all the six half-axes of an ellipsoid can be of different lengths, and thus brings the possibility of creating non-symmetric ellipsoids. Figure 6.8 depicts an example of this Stitch-Piece-to-Ellipsoid construction. We can also find some extreme examples of
CHAPTER 6. PRIMITIVE METHOD

6.4 Modification operation in Primitive module

6.4.1 Primitive object selection

To modify an Primitive object, e.g., to do movement or transformation operation on it, we need to select it first. This selection operation involves selecting the whole control frame or part of it. For the complexity of modifying primitive objects, a Selection Loop mechanism is used for the primitive object selection. Different selection results of a primitive object can be obtained by sequentially click on it.

- Column object selection:
  
  There are four selection status for a Column object. These selection status can be activated repeatedly in the order of Non-selected, All-selected, Bottom-surface-selected and Top-surface-selected. An example of this column object selection can be found in Figure 6.9.

- Ellipsoid object selection:
  
  There are six selection status for an Ellipsoid object. Sequentially, these six status of Non-selected, All-selected, Horizontal-control-frame-selected, Vertical-axis-selected, Bottom-control-point-selected and Top-control-point-selected can be obtained by repeat mouse left button clicks on a ellipsoid model. Figure 6.10 illustrates this Ellipsoid object selection loop with the Non-selected status not displayed in it.
6.4.2 Column modification

- **Column object movement:**

  In 2D workspace, if a column object is selected and the selection status is "All-selected", the entire object can be horizontally move to any place by a press-down-move operation starting from a non-control-point position on the design canvas. In 3D workspace, with the "All-selected" status kept, the object can be moved vertically also by the press-down-move operation.

- **Column object horizontal surface 2D movement:**

  We can move a selected bottom or top surface of a column object in 2D workspace, and the height/thickness of the object will not be changed. This is often used to align a designated control point of one horizontal surface to a control point of another horizontal surface without changing both the surface shapes. Figure 6.11 (a)-(b) shows an example of this surface 2D movement operation.
Figure 6.11: **Column object surface movement in 2D and 3D workspace.** In (a)-(b), the top surface of a transformed *Frustum Pyramid* is horizontally moved in 2D workspace by aligning a control point of the top surface to a control point of the bottom surface. This operation changes a skew side face to a vertical one. In (c)-(d), the bottom and top surfaces are vertically moved in 3D workspace. This operation makes an ellipse-section *Frustum Cone* switch its two horizontal surfaces up side down. The height and elevation of this column object are also changed.

Figure 6.12: **Column object surface transformation.** The column object *mutual-conversion* is showed in (a)-(c). A polygon-section *Prism* (a) is transformed to a *Frustum Pyramid* (b), and sequently changed to a *Pyramid* (c). In (d)-(f), by modifying the top surface of an ellipse-section *Cylinder* (d), we obtain a round top surface *Frustum Cone* (e) and a funny column model (f).

- **Column object height/thickness adjustment:**

  We can also change the elevation of the bottom or top surface of a column object, with the result of changing the height or thickness of it, in 3D design environment by the simple and efficient *press-down-move* operation. This function is typically used to change the height/thickness of a Column object. Figure 6.11 (c)-(d) illustrates an example of this height/thickness adjustment operation.

- **Column object surface transformation:**

  Using the *press-down-move* operation, we can transform the bottom and top surfaces of a column object in 2D workspace by manipulating its appropriate control points. This
6.4.3 Ellipsoid modification

- Ellipsoid object movement:

Similar to the object movement operation used for Column object, we can select an Ellipsoid object to the “All-selected” status and move it in 2D or 3D workspace using press-down-move operation. A small difference is we can have more freedom when we move an entire ellipsoid object in 2D workspace. In that case, we only need to select that ellipsoid object to any status except “Non-selected”.

- Ellipsoid object horizontal control frame elevation alteration:

After making the selection status of an Ellipsoid object to “Horizontal-control-frame-selected”, we can move the horizontal-control-frame of the object, along with the center point of the ellipsoid moved, vertically in 3D workspace. This changes the lengths of the two half axes of the upper-half and lower-half ellipsoids positively or negatively at the same time, with the bottom and top control points unmoved. This function
CHAPTER 6. PRIMITIVE METHOD

Figure 6.14: **Ellipsoid object vertical axis length adjustment.** This figure shows how to symmetrically change the vertical axis of an *Ellipsoid* object.

![Ellipsoid object vertical axis length adjustment](image)

Figure 6.15: **Ellipsoid object bottom/top control point movement.** This figure shows that by vertically moving the top control point of an ellipsoid object, we can change an ellipsoid (a) to a half ellipsoid (c), and finally make it a half shell (d).

![Ellipsoid object bottom/top control point movement](image)

is quite useful when we want to change a convex ellipsoid to a concave ellipsoid, e.g., to model a bowl based on a sphere. Figure 6.13 shows an example of this *horizontal control frame elevation alteration* operation.

- **Vertical axis length adjustment of Ellipsoid objects:**

  We can simultaneously prolong or shorten the two vertical half-axes of an *Ellipsoid* object when we select it to the status of "Vertical-axis-selected". Unlike the "Horizontal-control-frame-selected" status, under this "Vertical-axis-selected" status, the press-down-move operation won’t change the position of the ellipsoid center point, but move the bottom and top control points vertically up or down at the same time. This is often used to do some semi-symmetric operation, e.g., to change a sphere to an axis-symmetric ellipsoid. Figure 6.14 shows an example of this *vertical axis length*
Figure 6.16: **Primitive model 2D center modification.** After selecting the top surface of a square prism, we (a) move the mouse cursor to the center control point, and (b) press-down-move mouse into the 1st quadrant, or (c) press-down-move mouse into the 2nd quadrant, or (d) press-down-move mouse into the 3rd quadrant, or (e) press-down-move mouse into the 4th quadrant.

- **Bottom/top control point movement of Ellipsoid objects:**

After selecting an ellipsoid to the status of "Bottom-control-point-selected" or "Top-control-point-selected", separately moving the bottom or top control point gives us more freedom on modifying an ellipsoid. This function can let us create non-symmetric ellipsoid in the vertical direction. An example of this bottom/top control point movement operation is illustrated in Figure 6.15.

### 6.5 Modeling aspects in **Primitive** module

#### 6.5.1 **Primitive** object 2D center modification

Primitive object 2D center modification function is used to uniformly control a primitive object in 2D domain. After a primitive object is selected, we can press-down-move the center control point to different quadrants to obtain different modification results. Figure 6.16 illustrates an example of doing this 2D center modification to the top surface of a square prism.

#### 6.5.2 **Column** object horizontal surface selection auto-switch

When one of the two horizontal surfaces of an **Column** object is selected, its selection status can be automatically changed to "Bottom-surface-selected" or "Top-surface-selected" based on the surface position during the vertical movement process. It is easy to use this feature to
change the elevation of one selected column horizontal surface along with another horizontal surface untouched.

When the selected surface is vertically moved to the exact elevation of another horizontal surface, the thickness of the column object turns to be zero. In this case, only the projection of the object frame will be displayed at that elevation and thus clearly indicates the critical plane.

After the selected surface crosses the critical plane, both of the selection status of the two horizontal surfaces are automatically switched. The showed control frame is changed at the same time to let users always know the correct vertical direction of the workspace. This is quite useful especially when the global design environment is not in the viewport. A horizontal surface vertical movement of an ellipse-section Frustum Cone is showed in Figure 6.17 to depict this column object horizontal surface selection auto-switch feature.

### 6.5.3 See-through and Footprint observation features for Ellipsoid objects

When we want to observe the inner axes of an ellipsoid, our attention is apt to be distracted by the object grids, no matter we display the model in the translucent or transparent mode. To solve this problem, a See-through function is provided in our system. When a user press down and hold the left mouse button, the grids of the selected ellipsoids will disappear and thus let the user clearly see the axes inside. After releasing the mouse button, the object grids will be displayed again.

Another observation feature, the Footprint of an ellipsoid, is provided to help users know
the original status of the ellipsoid objects they are modifying. The Footprint of an ellipsoid is its original control frame displayed in gray color before the current modification operation. One can easily compare the original and current control frames of a being modified ellipsoid based on this function. The Auto-snap function can also be applied to the Footprint feature and this makes the modified object go back to the original status possible.

Both of these two observation features can be used in 2D and 3D workspace. Figure 6.18 shows an example.

### 6.6 An example of modifying Ellipsoid objects based on half-axis observation

Here we show an example of modifying Ellipsoid objects based on half-axis observation to illustrate the typical modification operation on Ellipsoid models. By using press-down-move operation along with half-axis observation in 2D and 3D workspace, we gradually change a non-symmetric ellipsoid to a uniform sphere. Figure 6.19 illustrates this modification processing.
Figure 6.18: **See-through and Footprint observation features for Ellipsoid objects.** These features are illustrated by an ellipsoid object in this figure. The 2D selected ellipsoid, its see-through result and footprint are showed in (a). The 3D corresponding views are showed in (b).

Figure 6.19: **An example of modifying Ellipsoid objects based on half-axis observation.** (a)-(j) depict the modification processing of gradually changing a non-symmetric ellipsoid to a uniform sphere.
Chapter 7

Design Examples

Our goal is to provide designers a handy and useful tool for creating 3D architectural models in the conceptual design stage. No matter what the complexity of the desired model is, we want the modeling process to be as simple, intuitive and efficient as possible. In this chapter, we show some architectural 3D models finished by using our prototype in the following pages to illustrate the capability of this modeling system.

First, we show some simple objects and architectural models in Figure 7.1 and Figure 7.2. Correspondingly complex architectural entities can also be modeled by using our system. Besides the Fallingwater Villa model showed in Figure 1.1, some of them are absolutely new designs beginning with drawing sketches on the virtual canvas of our modeling design environment (Figure 7.3 and Figure 7.4), and some of them are created by taking architecture photos as reference (Figure 7.5 and Figure 7.6).

These design examples showed in this chapter indicate that our prototype sketch-based modeling system has the ability to create architectural models with various complexity. From simple objects to architectural details, users can create them without inputing a single parameter. Sketch mode brings the traditional pencil and tracing paper into a computer-aided design system and plays the role of guiding the modeling process as reference. With this modeling method, we integrated the good features of free hand sketches applied in designs and the advanced 3D modeling capabilities provided by computers.

Based on the experience of finishing these models, we found that a user can start doodling without having a specific goal in mind, as if scratching on papers. Traditional modeling systems require a specific goal and careful planning before starting to work on the model, which can hinder the creative process. Using our system, designers can create models by
drawing casual shapes, and then gradually deform and refine them guided by sketches and their serendipity, which are very important for creative work.

Figure 7.1: Some simple models created using our system. (a) depicts some typical rudimental shapes we can obtain in the system. An imaginary flower and a goblet are showed in (b) and (c) respectively.

Figure 7.2: Two examples of modifying simple architectural models for conceptual designs. These coarse yet impressive models can clearly show architects' initial imagination towards some specific projects, and can be used as the basic of performing detailed designs.
Figure 7.3: A conceptual design of a museum finished using our system. (a) shows the 3D translucent model while the 2D top view is showed in (b). (c) depicts a detailed perspective view of the entrance. Using the modification functions provided by our system, creating typical slope roof is an easy job in the modeling process. Different observation modes can let architects control their designs easily.
Figure 7.4: A courtyard landscape conceptual design obtained using our prototype. (a) illustrates the integrated design/draw environment provided by our system, while (b) shows the 3D result. After obtaining the models in our system, we exported them out to an "obj" file and loaded it into 3ds MAX. (c) and (b) depict the top view and perspective scene rendered by 3ds MAX. This example shows the file quality of the exported models is good enough for transferring 3D data into industrial modeling tools.
Figure 7.5: A *London Tower Bridge* model created using our prototype based on a photo. (a) and (b) illustrate the 2D and 3D design results, while (c) shows a perspective rendering obtained in *3ds MAX*. Here the *Extrusion* and *Primitive* models are combined together to create complex architectural entities.
Figure 7.6: An model of the new *St. Anne’s Church, Bukit Mertajam, Malaysia*. (a) depicts the 3D model obtained using our system and (b) illustrates the crisscross top view. (c) shows a perspective rendered by *3ds MAX*. Almost all the geometric shapes in this example are *Primitive* objects. This shows sometimes the *Primitive* method is a very efficient way for creating architectural models.
Chapter 8

Conclusion and Future Work

In this report, we present a prototype sketch-based modeling system for the practical application on architectural conceptual design. Traditionally, architects use pencils to draw sketches on papers to conceive design schemes, express their thoughts and make certain the initial ideas to the final designs. Pencils and papers give architects a lot of freedom on their design work, and now in the architectural design industry, this traditional design method is still widely used.

Besides all the advantages of this traditional pencil-paper design method, one drawback is that it needs architects to manually draw perspective views to express their designs. This is a cumbersome work for an architect, especially when many perspective pictures from multiple view angles are needed. To aid architects finish this drawing task, some CAGD software, such as AutoCAD, 3ds MAX, SketchUP, etc., provide powerful functions to fulfill the modeling needs. But, these software don’t provide an virtual design environment completely conforming to the traditional pencil-paper design habits. Our prototype modeling system is provided to fill this gap.

We provide a virtual design environment which integrate the sketch/draw modes together to let architects create 3D models using their traditional pencil-paper experience. Our system uses good hardware features of the Tablet PC and focuses on interaction between the user and the computer. Using this system, a designer can sketch on the virtual canvas with the same feeling of drawing on the physical paper, along with the function of creating 3D objects based on their sketches.

Two categories of model creation, i.e., the Extrusion and Primitive modules which are widely used in architectural entities, are supported by our prototype system. As the basic
CHAPTER 8. CONCLUSION AND FUTURE WORK

modeling tool, the *Extrusion* module makes the design process conform to traditional design habits. Polygonal or curved base contours can be extruded up to create 3D models. As a very useful supplement to the *Extrusion* module, the *Primitive* module makes the creation of the *Column-like* or *Ellipsoid-like* models an easy job. All the created models can be further modified or moved to desired space position.

The sketch-based modeling problem is an interesting research direction with very wide application prospect. A mature geometric modeling tool is a very complex system and we can only make some limited attempts in this prototype. Many areas of our work can still be improved upon. We discuss below possible future works and the difficulties in solving them.

- **Unlimited sketches on models:**
  
  Currently, our prototype system only support sketching on *fiducial plane*—the horizontal plane at the elevation zero. In practical use, this should be extended to any surface on the models, because users may want to take an arbitrary surface on the models as a new fiducial plane to start their new designs. This need to bring the UCS(User Coordinates System) into our system.

- **UCS(User Coordinates System):**
  
  Our prototype system only support WCS(World Coordinates System) and all the computation is based on the system origin—the (0,0,0) coordinates. In practical use, this is not sufficient. UCS, the *User Coordinates System*, is need to locally control the model creation. All the UCS should be mapped back to WCS after the computation.

- **Multiple piece Beziér curve recognition:**
  
  In the current system, only one piece cubic Beziér curve recognition is supported. Although it can fulfill most needs in the modeling process, and our prototype system has already provided curve connection and smoothness mechanism to obtain continuous curves with multiple bending, it is convenient for users to get a recognized curve fitting their arbitrary sketches. More research on this fitting problem should be done in our future work.

- **Independent surface transformation of *Extrusion* models:**
Now, we can only simultaneously change the bottom and top horizontal surfaces of an Extrusion model. A single base contour is used for both of those two surfaces. In fact, we should give users more freedom on control the Extrusion models. This is quite useful on model modification, e.g., to change an initial extruded model into a sloped roof.

- **Model growth on Extrusion models:**
  Model growth function makes the design process a thinking process. Many practical modeling operations are based on this technique, e.g., to open a window on a wall or to erect a chimney based on a roof. More endeavor will be put into our future work to make this typical application a reality. A lot of techniques will be involved in it, such as Boolean Calculation and the application of UCS computation.

- **More detailed modification functions on Extrusion models:**
  Our future system should support more detailed modification functions. For example, we should let a user easily make a rigid surface intersection round based on sketch operations. Chamfering an object should be made an intuitive and direct action. These functions are very important derived from our basic system design philosophy of focusing on interaction.

- **More Primitive model modification freedom:**
  We should provide more freedom on modifying the Primitive models. This can bring more importance to the primitive module on creating basic parts. Advanced primitive model creation and modification techniques can greatly alleviate a designer's modeling effort and lower down the overhead computation consumption on time and storage.

- **More supported Primitive model creation:**
  More Primitive model creation functions should be added into our future system. Other than the Column-like and Ellipsoid-like primitive models are currently supported in the current prototype, the Tube-like, Donut-like, and Spring-like models, etc., should be included in our modeling system. This helps users create more diverse models in their practical design.

- **More utility functions:**
Except for the simple movement function provided by our prototype, more utility functions, such as copy, mirror, rotation, scale, group, etc., should be obtained in our future system. These useful functions can finally make our prototype a practical modeling tool.

We believe that our prototype does have great potential in commercialization. In the laptop market, it is a trend to widely provide hardware support of screen writing to portable computation in the very near future. The expense of obtaining a Tablet PC in one or two years can be similar to or less than a current normal laptop. This will eliminate the overhead of using Tablet PCs for companies and individuals. In the software industry, we can now hardly find a mature solution of performing architectural designs using Tablet PCs. With the good features of sketch-based modeling, the way of designing architectures on Tablet PCs conforming to traditional habits will be accepted by more and more architects.
Bibliography


