ArchiDNA:  

A Generative System for Shape Configuration 

Doo Young Kwon  

A thesis submitted in partial fulfillment of the requirements for the degree of  

Master of Science  

University of Washington  
2003  

Program Authorized to Offer Degree: Architecture
University of Washington
Graduate School

This is certify that I have examined this copy of a master’s thesis by

Doo Young Kwon

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final examining committee has been made.

Committee Members:  ___________________________
  Ellen Yi-Luen Do
  ___________________________
  Mark D. Gross
  ___________________________
  Brian R. Johnson

Date:  ___________________________
In presenting this thesis in partial fulfillment of the requirements for a Master’s degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with “fair use” as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature: ____________________________________________

Date: ____________________________________________
Abstract

ArchiDNA:
A Generative System for Shape Configuration

Doo Young Kwon

Chair of the Supervisory Committee:
Professor Ellen Yi-Luen Do
Department of Architecture

This thesis introduces a new approach to generating shape configurations. The approach derives from analyzing and implementing a certain design style into a computer. From this, the thesis investigates how a generative CAD software can be developed to embody a style and how this software can serve as a computational tool in design. This thesis shows two working systems ArchiDNA and ArchiDNA++. ArchiDNA introduces a simple generation process, called application process, that allows a designer to compose shapes using a set of operations. Using the process, ArchiDNA demonstrates that a certain design style can be programmed into a computer and a designer can generate the same stylistic shape configurations in the same style. Developing ArchiDNA++ explores the use of application process to support designers generating new configurations for their design. The advantage of this approach, as demonstrated in the two systems, is that a certain design style can be captured by a set of shapes and their operation rules. Moreover, unlike other CAD softwares, it allows an easy, fast and systematic process to generate stylistic shape configuration.
Table of Contents

List of Figures.................................................................................................................. iv
Acknowledgements ........................................................................................................ vii

Chapter One: Architectural Drawing & Style Analysis 1

1. Introduction .............................................................................................................. 1
   1.1. A Role of Architectural Drawing ................................................................. 1
       1.1.1. A Generative Device to Find a Building Form ............................... 1
       1.1.2. Computer Generated Drawing ....................................................... 2
       1.1.3. Generative Drawing Machine .................................................... 3
   1.3. Style Analysis ................................................................................................. 4
       1.3.1. What is Design Style? ................................................................. 4
       1.3.2. Style Analysis in a Drawing ....................................................... 5
       1.3.3. Style Analysis with a Computer .................................................. 6
   1.5. The Goal of the Thesis ................................................................................. 7
   1.6. ArchiDNA: An eisenman design machine ................................................. 8
   1.7. Outline of the Thesis .................................................................................... 9

Chapter Two: Analysis of Style and Generative Systems

2. Related Work.......................................................................................................... 11
   2.1. Analysis of Style .......................................................................................... 11
       2.1.1. Shape Grammar ........................................................................... 11
       2.1.2. Analysis and Generation of Style with Shape Grammar .............. 13
       2.1.3. Analysis and Generation of Style with Other Algorithmic Approaches ..... 20
   2.2. Generative Applications ......................................................................... 23
       2.2.1. Shape Grammar Interpreters ....................................................... 23
       2.2.2. Algorithmic Shape or Form Generation .................................. 27
   2.4. Discussion .................................................................................................... 31
       2.4.1. Analysis of style ........................................................................... 31
       2.4.2. Generative Applications ............................................................ 32

Chapter Three: An Eisenman Design Machine
3. ArchiDNA .............................................................................................................. 34
   3.1. Eisenman’s Biocentrum Building Design ......................................................... 34
   3.2. 2-D Operations ................................................................................................. 36
      3.2.1. Primitive Operations in ArchiDNA .......................................................... 36
      3.2.2. Application Process: applier-shape and base-shape .................................. 38
   3.3. 2-D Generation ................................................................................................. 39
   3.4. 3-D Operations ................................................................................................. 42
   3.5. 3-D Generation ................................................................................................. 43
   3.6 Summary ............................................................................................................ 44
   3.7 Discussion .......................................................................................................... 44

Chapter Four: ArchiDNA++ An End-User Programming Environment For Shape
Application

4. ArchiDNA ++ ........................................................................................................... 46
   4.1. ArchiDNA++ Environment ............................................................................. 47
      4.1.1. Tool Palette ............................................................................................... 47
      4.1.2. Drawing Window ...................................................................................... 47
      4.1.3. Four Control Palettes ............................................................................... 48
   4.2. ArchiDNA++ in use .......................................................................................... 49
   4.3. Shape Preparation Phase .................................................................................. 50
      4.3.1. Shape Creation ............................................................................................ 50
      4.3.2. Shape Attribute Control ............................................................................ 51
   4.4. Shape Configuration Phase ............................................................................... 54
      4.4.1. Define applier-shape(s) ............................................................................ 54
      4.4.2. Select base-shape or Draw base-shape ...................................................... 54
   4.5 3-D Conversion .................................................................................................. 55
      4.5.1. Set a Threshold ......................................................................................... 55
      4.5.2. Set 3D Mode ............................................................................................... 55
   4.6. Example: Generation of Sunflower Shape ....................................................... 57
      4.6.1. Shape Preparation ...................................................................................... 57
      4.6.2. Shape Configuration ................................................................................... 59

Chapter Five
List of Figures

Figure 1.1: Drawings for Peter Eisenman’s House Project (Eisenman, 1975) ........................................2
Figure 1.2: Computer generated drawings for Eisenman’s Groningen Video Pavilion Project (Eisenman, 1990) .........................................................................................................................3
Figure 1.3: (a) Plan Drawing for Biocentrum (Eisenman 1986) (b) 3D Model for Biocentrum (Eisenman 1986) .................................................................................................................................9
Figure 1.4: (a) 2D Eisenman-like drawing generated in ArchiDNA (b) 3D ArchiDNA model converted from 2D Eisenman-like drawing on VRML viewer, Cortona on Microsoft Internet Explorer .9
Figure 2.1: (a) Rules for a Standard Shape Grammar (b) A derivation of the rules (c) A result shapes generated by applying the rules ..........................................................................................12
Figure 2.2: (a) Rules for a Parametric Shape Grammar (b) A derivation of the rules (c) A result shapes generated by applying the rules ..........................................................................................13
Figure 2.3: (a) Some results of ice-ray grammar (b) Four rules for a shape grammar of an ice-ray pattern ........................................................................................................................................14
Figure 2.4: A derivation by ice-ray grammar ..........................................................................................14
Figure 2.5: Possible Palladian villa plans with Palladian villas grammar ..............................................15
Figure 2.6: A derivation of Villa Malcontenta using Palladian villas grammar ......................................16
Figure 2.7: One of massing rules (a) and detailing rules (b) for the prairie house grammar and a derivation of the rules (c) ..............................................................................................................17
Figure 2.8: Various results of the prairie house grammar ........................................................................18
Figure 2.9: A part of the rules for a Diebenkorn Ocean Park Grammar ................................................19
Figure 2.10: Steps in the generation of Diebenkorn’s Ocean Park No. 111 from the grammar ............19
Figure 2.11: (a) Generation rules for wall architecture (b) Configurations generated by these rules ....20
Figure 2.12: (a) Snapshot of a program, “Make Anthropomorphic Shape” (b) Various anthropomorphic shapes modified in the program. .........................................................................................21
Figure 2.13: (a) Prepared shapes from analyzing Miro’s painting (b) A final Miro composition generated by using the prepared shapes ................................................................................................22
Figure 2.14: Generation of a Palladian villa in PlanMaker system .......................................................23
Figure 2.15: Possible Palladian villa plans generated in PlanMaker system ........................................23
Figure 2.16: Various 2D labeled rules for standard shape grammar and the derivations for the rules ...24
Figure 2.17: Various 3D labeled rules for standard shape grammar and the results of the rules...........24
Figure 2.18: Screenshot of GEdit Interface ............................................................................................25
Figure 2.19: Screenshot of Shaper 2D Interface .....................................................................................26
Figure 2.20: Illustrations for using the result of Shaper2D in the design process (a) The generated result in Shaper 2D (b) Site planning with the result (c) Plan designing with the result

Figure 2.21: (a) Screenshot of 3D Shaper Interface (b) Screenshot of SGI Open Inventor Viewer to see the 3D result of 3D Shaper

Figure 2.22: The turtle commands to generate shape configuration in Logo Programming Environment

Figure 2.23: Snapshot of Design by Number Interface http://dbn.media.mit.edu/

Figure 2.24: Various Results of Design by Number

Figure 2.25: Snapshot of FormWriter Interface

Figure 2.26: Models of historic Islamic structures by students using the FormWriter

Figure 2.27: Snapshot of Processing Working Environment

Figure 3.1: (a) Biocentrum (Eisenman, 1996) (b) Diagram of DNA showing Amino Acids (c) Four distinct shapes in Amino Acids

Figure 3.2: Drawings for Biocentrum (Eisenman, 1996) (a) Plan (b) Axonometric View

Figure 3.3: Primitive Shape Operations

Figure 3.4: A square is attached to a line considering its location, angle, and size

Figure 3.5: Square is attached to four edges of the first square

Figure 3.6: Four Shape Operations with applier-shape to base-shape

Figure 3.7: Initial shape in ArchiDNA Interface

Figure 3.8: (a) Application the applier-shape G (ribbon) to the base-shape C (pentagon) (b) Application the applier-shape G to the base-shape A (arch)

Figure 3.9: ArchiDNA Second Shape Generation

Figure 3.10: 2D Eisenman-like drawing generated in ArchiDNA

Figure 3.11: 3D form generation in ArchiDNA (a) Calculating the area of two applied shapes A1&A2 and assigning heights (b) Comparing areas with a threshold and deciding up and down (c) Extrusion of small shape A1 upward and extrusion of large shape A2 downward (d) Extrusion of shape B with a user-defined height

Figure 3.12: 3D Eisenman-like Drawing Generated in ArchiDNA

Figure 3.13: Overview of ArchiDNA++ Operation
Figure 4.5: Various Created Shapes; a rectangle (□) in the middle of an edge marks an anchor-edge. ....51
Figure 4.6: Changing anchor-edge of applier-shape object (ribbon) and the result of applying it to a base-shape object (pentagon); dotted lines illustrates the default configuration that uses anchor-edge at top of ribbon. ........................................................................................................52
Figure 4.7: Change two attachable-edges of base-shape object (pentagon) to un-attachable edges and its result with an applier-shape object (ribbon). ........................................................................................................52
Figure 4.8: Fix three edges with angle (90°) and scale-factor (50%) and its result shows three applier-shape objects are not matched to the base-shape object. ..........................................................53
Figure 4.9: (a) Counter-clockwise base-shape object (b) Clockwise base-shape object with four applier-shape objects (A1-A2-A3-A4) ...........................................................................................................53
Figure 4.10: Various Shape Configurations in ArchiDNA++. ..........................................................53
Figure 4.11: 3D Mode: block-mode and wall-mode (a) 3D model generated in block-mode (b) 3D model generated in wall-mode. ..................................................................................................................55
Figure 4.12: (a) ArchiDNA++ 3D Model in VRML Viewer, Cortona (b) ArchiDNA++ 3D Model imported into formZ. ..........................................................................................................................55
Figure 4.13: Create three shapes (ribbon, triangle, and hexagon). ..........................................................58
Figure 4.14: Change shape attributes (attachable-edge and anchor-edge). ..........................................................58
Figure 4.15: First shape configuration for sunflower-like shape attaching ribbons to a pentagon .......60
Figure 4.16: Second shape configuration for sunflower-like shape attaching a triangle to the ribbon base-shape objects ..................................................................................................................60
Figure 5.1: ArchiDNA System Architecture. .............................................................................................62
Figure 5.2: Representation of a ribbon-like shape with points and edges (a) five points with 2D coordinates (b) five edges with anchor-edge mark [*], attachable-edge mark [+] , un-attachable-edge mark [-], and operation values [angle and scale] (c) the final display on the screen. ......63
Figure 5.3: Data Structure for a ribbon-like shape in ArchiDNA ..................................................................................64
Figure 5.4: A System Diagram for the Process of Shape Configuration .............................................................65
Figure 5.5: Converting a 2D shape (ribbon) to 3D VRML objects. .................................................................66
Figure 6.1: Representation of a ribbon space .................................................................................................69
Figure 6.2: Data structure for a ribbon space .................................................................................................69
Figure 6.3: 3D conversion of a ribbon space .................................................................................................69
Figure 6.4: Rules for a grammar of ArchiDNA application process .................................................................71
Figure 6.5: Optional Rules for a grammar of ArchiDNA application process .................................................................71
Figure 6.6: 3D form configuration of applying 3D applier-shape to 3D base-shape ........................................72
Acknowledgements
Chapter One

Architectural Drawing & Style Analysis

1. Introduction

1.1. A ROLE OF ARCHITECTURAL DRAWING

1.1.1. A Generative Device to Find a Building Form

Designers often draw shapes to investigate ideas about building forms in the schematic design process. Herbert argues that “study drawings” is a medium for designers to find formal design ideas. He states that the important characteristic of the study drawing is its graphic ambiguity. Unlike final plan drawings with details and specification, study drawings provide a variety and fluid formal design ideas to the designer. Thus this kind of drawing is useful for designers to explore a building form in a schematic design process. The ambiguity of drawing could be obtained with irregular overlays of various colors, in shapes, spaces, lines, or in images with any specified degree of irregularity. Fraser and Henmi (1993) also argue the importance of ambiguity and irregularity in drawings. Architects use “design drawings” to find design idea. They draw shapes from their minds’ eye or preconceived images. Then the designers develop further a drawing by combining these previously generated shapes. This process is repeated until the drawing get closer to the designer’s preconceived images. Through this repetitive generation process, the designer finds or develops a formal design idea.

Peter Eisenman, a contemporary American architect made a specific statement about drawings as a generative device. He stated that “What I do is set up a series of ideas or rules or strategies and draw into those, trying to find some form in those ideas (quoted in Herbert, 1993).” He uses drawing in his design process to find the relationship of the formal to the conceptual. He intentionally manipulates and utilizes the effects of drawing as an explicit design problem.
For Eisenman, drawing is a way of both searching for a form and idea, and also explaining how the form and idea can be manipulated. Figure 1.1 shows a set of drawings from Peter Eisenman’s House project. He repeatedly adds lines on the previous drawings and makes it more complex. This series of drawings illustrate how he uses drawings as a generative device to explore formal design idea.

![Figure 1.1: Drawings for Peter Eisenman’s House Project (Eisenman, 1975)](image)

### 1.1.2. Computer Generated Drawing

Architects are increasingly using computers to generate 2D and 3D drawings in the design process. They use a computer not only to represent the final product but also to explore architectural form during the schematic phase of design. Kolarevic (2000) surveys some different approaches architects use the computer to find a building form in contemporary architectural design. He called these approaches digital architectures (Kolarevic, 2000). In this case, various computer software systems are used to manipulate or generate forms for architectural design. For example, Greg Lynn (1999) uses animation software to generate building form. He visualizes his design concept with animated 3D objects and develops to an architectural form.

Peter Eisenman is also widely considered as a designer of digital architecture (Kolarevic, 2000). He believes the computer-generated drawings are useful for exploring the building form possibilities. He uses computer algorithms to generate drawings for formal design idea in design process. Figure 1.2 shows computer-generated drawings for Eisenman’s Groningen Video
Pavilion (Eisenman, 1996). Different from the drawings for House project [Fig. 1.1], he uses a simple CAD functions (copy, translation, and rotation) which manipulate shapes. In stead of drawing with a pen, he used those operations using a mouse in a CAD system. This computer drawing process is also repeated to generate a series of alternative drawings [Fig. 1.2].

![Figure 1.2: Computer generated drawings for Eisenman’s Groningen Video Pavilion Project (Eisenman, 1990)](image)

1.1.3. Generative Drawing Machine

The previous sections describe that architects generate drawings to explore formal design idea. They use a conventional drawing tool (pen and paper) or a CAD program. Regardless which drawing medium, they represent and develop their visual idea repeatedly until finding a building form. We believe a design tool which allows designers to quickly create drawing alternatives might be useful for design. Even though current CAD softwares support basic drawings, it is not easy to learn and control these operations. Even if a designer already knows about a CAD system, it can be time consuming and labor-intensive to produce proposed drawings that support a formal design idea. This is one of reasons why a designer uses a computer just to present a final product.

Computers can be useful for a designer to get results in an effective way. As explained before, using computation, a computer can generate various drawing results from a designer’s simple input. Therefore, various research efforts such as shape grammars and algorithmic shape generations have been invested in this approach. They describe how a computer can algorithmically generate multiple formal ideas in an early design phase.
These generative systems involve the concept of computation that uses input to produce output. A computer knows how to produce with the input following an algorithm (Turing, 1936). Using the concept of computation, a generative drawing machine can be developed with shape manipulation algorithms. When a designer input certain shapes, the computer generates shape configurations following the algorithms defined by the designer. A designer can then generate different shape configurations by repeating this generation process in a design process. This generation process is similar to Eisenman’s drawing method, which sets up “a series of ideas or rules or strategies” and draws from them, “to find some form in those ideas. (Eisenman, 1999)”

The common goal of these systems is to generate drawings or describe the drawings using computer algorithms. In other words, a designer uses this system to investigate formal design idea and explain the generated form with the used computer algorithms. The designer eventually uses the algorithms as a design tool to create building form.

In this thesis we are investigating two important issues: to define the algorithms for the shape generation and to make the generation process transparent for the designer. The shape generation algorithm can be developed for designers to create complicated shape patterns or to manipulate the existing shapes. Moreover, the algorithms can be embedded with a certain design knowledge that controls the shape generation or manipulation. Drawing is a result of thinking in the design process. Therefore the drawing machine should provide an interface for designers to control and manipulate those algorithms.

1.3. STYLE ANALYSIS

1.3.1. What is Design Style?

Style has been explained in several ways. Gombrich (1960) and Simon (1975) defined style as a way of doing things. A style is produced when people select or produce a particular alternative or process to find a solution. Style usually expresses characteristics of an individual or a group of people for particular ways of thinking and doing things.
Design style is a kind of design knowledge that characterizes a particular design artifact or a group of design work (Shapiro, 1961). A design style can be described as consisting certain elements and ways to compose those elements. For example, when we look at architectural drawings or buildings, we can recognize their forms and qualities as displaying a certain style. The formal properties such as shape, color, arrangement, texture, size, and orientation are visual elements for recognizing a style. In another words, a design style means that the features of an object as well as the relationships between the components of the object can be recognized explicitly. Therefore a style can be described in terms of the common components of a design and how they are arranged.

Many architecture schools teach various design styles in the design. Studio students learn how to develop their design knowledge by following other designs styles. They can create and manipulate design elements looking at the other designs. It can help the designer to understand the certain design style. Therefore, the designer can apply the understanding to his or her own design.

1.3.2. Style Analysis in a Drawing

The design style of architectural drawings can be analyzed through the identification of the drawing elements and their relationships. Architectural drawings usually represent architectural vocabularies such as walls, doors, columns, rooms, and so on. These architectural components are used as formal elements to analyze architectural drawings. For example, Stiny and Mitchell analyzed the plans of Palladian villas to define how the architectural component can be generated or arranged to create floor plan of the style of the Palladian villas. However, some drawing types (study drawings or design drawings) do not exactly represent architectural components. Architects use these drawings as a visual tool to find a building form in a schematic design process. These drawings might have form elements such as patterns and shapes. Using the form elements, a style of the drawing can be analyzed and described. For example, Stiny (1972) describes a stylistic pattern of the Chinese lattice window as a set of rules. Kirsch and Kirsch (1986) analyzed and described a style for Richard Diebenkorn’s abstract painting. Even though these are not architectural drawings, they show that a style can be defined
as consisting of elements and the relationships. Likewise, style in architectural drawings can be represented by a set of shapes and their relationships. The shapes can be recognized visually from the drawing. The spatial relationships can be represented with basic transformations (translation, rotation, and scaling) in a Cartesian coordinate system.

1.3.3. Style Analysis with a Computer

As explained in the previous section, a design style is a kind of design knowledge which can be learned and used in design process. A design style is recognized with formal properties of a drawing. The advance of computer technology makes it possible to analyze a design style. For example, a design style in a drawing is studied in the form of computer algorithms that manipulate the design components (Kirsch, 1998). Char and Gero (1999) identified common shapes in a set of architectural drawings to define a design style. They used basic transformations of computer graphics such as translation, rotation, scaling, and mirroring to represent the shapes and the relations. The purpose of these studies supports a designer to understand and learn a design style and develop his or her own style using a computer.

Since Stiny and Gips introduced shape grammar as an idea of using a set of rules to represent a painting (Stiny & Gips, 1972), many systems have demonstrated how a computer can generate shapes with certain graphic or architectural styles using the shape grammar. Much work in shape grammar has analyzed specific design styles with a set of rules and then showed how those rules can be used to generate new designs in that style (Knight, 2000). Separately from the shape grammar have been the studies using computer graphic algorithms. For example, Kirsch and Kirsch (1986) defined the style of the Spanish painter Joan Miro. They describe the common shapes in Miros drawings with algorithms which can manipulate the shapes. Hersey and Freedman (1992) also developed the split algorithm that divide a square horizontally or vertically for Palladian design style. This simple algorithm generates various building plans in the Palladian style. These systems show a design style can be defined into a set of algorithms and a designer can develop a new design style or modify an existing style systematically.
1.5. THE GOAL OF THE THESIS

This thesis examines how a generative CAD software program can be developed to embody a style and how this program can serve as a computational tool in design. It discusses about a design style in architectural drawing, a set of algorithms (or operations) to generate drawings in the design style, and a generative system to use the algorithms. This thesis has two main goals.

The first goal is to program a design style into a computer system. A design style is an important kind of design knowledge. If a design style can be programmed into computer algorithms, it can help a designer to understand and learn about this particular design style with a generative system. The system enables a designer to easily investigate formal design idea by interacting with the system.

A design style can be defined in several ways. The thesis investigate a design style in particular drawings. This thesis chose to start with a design style represented in Peter Eisenman’s drawings for Biocentrum Building. The drawing is composed with common drawing features of which composition can be geometrically described. The idea is that if we can develop an algorithmic process to generate the drawings, then we may conclude that a computer regenerates drawings of the same style using the algorithms. This thesis proposes a generative system, ArchiDNA to generate drawings of Eisenman’s style. This system demonstrates that certain style can be programmed as a set of operations and how a designer can use it to make Eisenman-like drawings.

The second goal of the thesis is to further generalize the algorithms previously defined for Eisenman’s design style, and develop a generative system that enables designers to use and control the generation process for their design. For this, this thesis explores the development of a system for designers to use the generation process of Eisenman-like drawing. This approach can be seen as extension of a CAD program. The thesis proposes an easy-to-use generative system, ArchiDNA++. There are currently many CAD softwares. However, it is still difficult for designers to learn the systems and use in the schematic design process. ArchiDNA++ is a working system that provides an interface. A designer can specify a design style with the use of
operations and to generate new drawings for design. The system also allows a designer to easily investigate formal design ideas.

1.6. ARCHIDNA: AN EISENMAN DESIGN MACHINE

ArchiDNA is a system that generates 2D and 3D shape configurations in the style of Peter Eisenman, in particular, in the style of his Biocentrum building. The system provides designers an easy and powerful method to quickly generate shape configurations.

Figure 1.3 shows Eisenman’s original 2D drawing and 3D model image for Biocentrum building design. Figure 1.4 shows a 2D shape configuration and its 3D form generated by a designer using ArchiDNA. Comparing these two drawings, we can see ArchiDNA generates 2D shape configurations and 3D form in the same style. This demonstration shows that a style can be expressed as a computer program and a designer can use ArchiDNA to generate shape configurations with this particular style.

This initial experiment of ArchiDNA led to the development of a second working system, called ArchiDNA++. This system uses the same process that generates drawings in the Eisenman’s style. However, it generalizes the generation process and supports a designer to manage the process. ArchiDNA++ enables expressing a designer’s own style by allowing the designer to create new shapes and control the operations. Unlike other CAD programs, ArchiDNA++ is a generative system that has a set of operations for a certain design style. A designer creates a number of shape configurations in all variations within a style.
1.7. OUTLINE OF THE THESIS

The rest of this document is divided into six parts. Chapter Two describes related work, including example applications of style analysis and generative systems for design. Chapter Three describes ArchiDNA to demonstrate how one defines a certain design style with a set of operations and generate drawings in the same style. Chapter Four explains the ArchiDNA++ software that allows a designer to control the operations and generate drawings. Chapter Five
explains the computational implementation of the ArchiDNA++. Finally, Chapter Six concludes with a discussion and identifies topics for future work.
Chapter Two

Analysis of Style and Generative Systems

2. Related Work

This chapter establishes background and underlying principles for this thesis. This chapter has three main sections. The first section reviews some analysis applications for design style. The second section describes some generative systems for design. The third section discusses with the proposed system.

2.1. ANALYSIS OF STYLE

2.1.1. Shape Grammar

Stiny introduces shape grammars to describe visual shape compositions (Stiny, 1985). A shape grammar has a set of rules which replace one part of shapes with another shape. This simple substitution process has been used to describe a certain design style and its design works into a set of rules. In other words, a shape grammar defines a certain design style into form of computer algorithms that manipulate the design components. Several systems have demonstrated using this shape grammar how a computer can generate shapes with certain graphic or architectural styles. Much work in shape grammar has analyzed a certain design style with a set of rules and showed how the rules can be used to generate new design in that style (Knight, 2000).

Stiny introduce two types of shape grammars: standard (non-parametric, basic, or standard) shape grammar and parametric shape grammar (Stiny, 1985). Standard shape grammars have a set of rules [Fig. 2.1a]. In these rules, the shape on the left side of the arrow determines which part of the shape will be replaced. The shape on the right side of the arrow decides the shape
that indicates this part. Following the sequence of rules, shapes are manipulated recursively [Fig. 2.2b]. The rules are applied recursively to an initial shape to generate an interesting drawing [Fig. 2.2c].

A parametric shape grammar also has a set of rules that specify how shapes replace sub-shapes of a composition [Fig. 2.2a]. However, it uses a parameter for shape manipulation. Shapes have proportion parameters, and the values of the parameters are left unassigned. When the rule is applied, the values are assigned by the rule. This parametric shape grammar creates shapes more variously rather than the standard shape grammar. Figure 2.2c shows the different result of the parametric shape grammar from the result of standard shape grammar [Fig. 2.1c].

These two shape grammars have been used for analysis applications for design style. Specially, next section explains some analysis applications for designs style using the parametric shape grammar. Then, this chapter also explains shape grammar interpreter systems that generate 2D and 3D drawings using standard shape grammar.

Figure 2.1: (a) Rules for a Standard Shape Grammar (b) A derivation of the rules (c) A result shapes generated by applying the rules
2.1.2. Analysis and Generation of Style with Shape Grammar

Ice-ray grammar is a parametric shape grammar that describes and generates instances of a Chinese lattice design style [Fig. 2.3] (Stiny, 1977). This grammar captures the compositional principle of lattice designs into a set of rules. It shows how to generate existing lattice designs as well as new designs in the same style.

This ice-ray grammar generates various patterns in the Chinese lattice design style [Fig. 2.3a]. Looking at Chinese window grilles, Stiny identified four rules for the ice-ray grammar [Fig. 2.3b]. The rules subdivide a shape by inserting a straight line. Figure 2.4 shows a derivation to generate a pattern starting from a rectangle shape using the rules. The rectangle is divided into two pentagons, and each pentagon is divided further into a triangle and pentagon. These subdivisions are applied recursively and generate a pattern in the Chinese lattice design style.
Figure 2.3: (a) Some results of ice-ray grammar (b) Four rules for a shape grammar of an ice-ray pattern

Figure 2.4: A derivation by ice-ray grammar

The next application is Palladian villas grammar (Stiny and Mitchell, 1978). Stiny and Mitchell defined a series of rules for villas designed by the sixteenth-century architect, Andrea Palladio. Figure 2.5 illustrates some villa plans described with the rules. These rules are more complicated than the previous ice-ray grammar. The Palladian villas grammar initiated more ambitious and complex parametric shape grammars for architectural styles that continue today. The Palladian villas grammar focuses on describing architectural plans that consist of walls, space, window, and entrance. It has 72 production rules that generate the original villa plans as well as new ones in the Palladian style. With these rules, The Palladian villas grammar generates the original building plan in detail following several stages.
Figure 2.6 shows a derivation to generate Villa Malcontenta with Palladian villas grammar. It starts from a single point, which shows a location of the plan on a site. A grid with rectangles is used as initial layout dimensioned in accordance with the proportions of the original Villa plan. These aspects of spatial arrangement control all subsequent stages of plan generation. The grid is used for generating external walls and rectangular spaces composed to form rooms in the plan. After that, the principal entrances and columns are added, and then windows and doors are inserted in walls to complete the plan.

*Figure 2.5: Possible Palladian villa plans with Palladian villas grammar*
Figure 2.6: A derivation of Villa Malcontenta using Palladian vials grammar
Koning and Eizenberg (1985) developed a 3D parametric shape grammar for Frank Lloyd Wright’s prairie house. They used 99 production rules, including 18 rules to arrange major cubic masses and 81 rules to add details to the masses. Figure 2.7a shows one of the massing rules. It extends one mass by attaching another mass to right side of the existed mass. Figure 2.7b illustrates one of detailing rules. It adds a terrace object to an existed building.

Figure 2.7c shows the steps of a derivation in part for the prairie house grammar. The house design starts from the fireplace and is organized around it. Then, a living zone is located around the fireplace creating a core unit. From this, the prairie house plan is composed with butterfly-shaped extensions of the core unit. The house plan’s basic composition is completed with named function zones such as living and service areas and porches and bedrooms. To decorate the house, rules add terraces, a basement, and a second story. The final rule completes the generation of the prairie house by adding a roof and chimney. Figure 2.8 illustrates some variations of the Frank Lloyd Wright’s prairie house using the rules.

![Figure 2.7: One of massing rules (a) and detailing rules (b) for the prairie house grammar and a derivation of the rules (c)](image)
Shape grammars are also used for describing the structure of a painting style. Kirsh and Kirsh (1986) described the patterns in the paintings of Richard Diebenkorn. They modified the parametric shape grammars to develop their grammar for a structure of painting. Figure 2.9 illustrates the part of the 42 production rules of the grammar. Most of rules are to subdivide regions of painting similar with the ice-ray grammar. Figure 2.10 illustrates a linear structure of the composition for Diebenkorn’s Ocean Park No. 111 by applying a sequence of some rules. Starting from applying Rule 7 to a rectangle, the selected rules are applied recursively until to get a final drawing.
Flemming (1990) presented grammars for various architectural languages: wall architecture, mass architecture, panel architecture, layered architecture, structure/infill architecture, and skin architecture. His main goal was to support students to learn architectural composition. He used shape grammars to define the rules indicate how architectural elements can be placed in the space. For example, Figure 2.11a shows a grammar for wall architecture. The rules of this grammar show how a wall attaches to another wall in space. Figure 2.11b illustrates some wall type configurations generated by applying the rules repeatedly. Likewise, other architectural languages are described with a set of rules. These grammars are used to teach students about architectural languages especially for the composition with a computer. Students use the
grammars to learn about the languages and then modify the grammars to generate their own design.

Figure 2.11: (a) Generation rules for wall architecture (b) Configurations generated by these rules.

2.1.3. Analysis and Generation of Style with Other Algorithmic Approaches

Kirsch and Kirsch (1998) describe the style of Spanish painter, “Joan Miro, who, in 1940 – 1941, painted a series of 23 gouaches called Constellations, which are complex allover
compositions based on suggestive and imaginative biomorphic forms.” The previous section described a grammar for Diebenkorn’s linear composition (Kirsch and Kirsch, 1985). However, in this case, they used a different approach because they found shape grammars are limited for representing shapes in Miro’s style.

Without using shape grammars, they analyzed Miro’s Constellation series paintings focusing on shape in studying Miro’s style. They scanned Miro’s drawings and manually traced the outlines of the shapes. Through these processes, they define a shape class for Miro’s paintings. Then they made several programs, each of which controls each shape of the shape class. Each shape is modified in various ways such as horizontal and vertical stretching, resizing, and rotating. For this, they developed several programs for some shapes. Figure 2.12a illustrates the interface of a program, “Make Anthropomorph Shape”. This program is developed in Macintosh Common Lisp. This program allows a designer to change the anthromorph shape by controlling a set of slide bars. Figure 2.12b illustrates the variations of the anthropomorphic shape generated in this system. Figure 2.13a describes a set of prototype shapes prepared through the previous steps. Finally, these shapes are synthesized into the Miro composition by being translated, rotated, and stretched [2.13b].

Figure 2.12: (a) Snapshot of a program, “Make Anthropomorph Shape” (b) Various anthropomorphic shapes modified in the program.
Hersey and Freedman (1992) developed a computer program, PlanMaker to generate possible Palladian building plans. They use a simple split system that splits a rectangle horizontally or vertically and resplit the previously split small rectangle again. This split system consists of three characteristics: a split direction (horizontal, vertical, or both), room number and split ration. Number defines the number of a room, which will be split in the next step. The split ration defines the relative proportions of the next room. These simple split system generates horizontally symmetrical and modular Palladian’s villa plans.

Figure 2.14 shows the five stages of sequential splits to generate the plan of Villa Valmarana at Lisiera in PlanMaker system. First stage illustrates the initial rectangle is split horizontally into three new rooms. Each new room is divided vertically again following a specific ratio again in the next stage. Following this simple splitting rule, the PlanMaker generates the original Palladian plan. Moreover, PlanMaker allow users to generate new drawing plans in the Palladian style following the split system. Figure 2.15 shows a variation of possible Palladian plans.
2.2. GENERATIVE APPLICATIONS

2.2.1. *Shape Grammar Interpreters*

As explained in the previous section, shape grammars have been used for describing a design style with a set of substitution rules. Shape grammars also have been used for generating 2D and 3D drawings. Shape grammar interpreters are programs that allow users to make rules for standard shape grammar and generate 2D and 3D drawings. A user arranges two shapes using simple operations (translation, rotation, scaling). The arranged shapes define the substitution rules by using an arrow and labels. Labels are symbols that say how to apply this rule in a
derivation. Figure 2.16 shows three labeled rules and derivations for each rule. The labeled rules show one square on the left side of the arrow determines which part of the shape will be replaced. The arranged two shapes on the right side of the arrow decide the shape that indicates this part. Considering the position of the label, each rule generates different a result. This simple rule is also applied to three dimensions. Figure 2.17 describes the 3D labeled rules and their results. In the similar way of the previous 2D rules, the 3D rules have 3D objects are separated with the arrow and each object has a label. These shape grammar rules are used for developing shape interpreter programs that allow a designer to create the rules and generate the result of the rules. Next explain recently developed three shape interpreters (GEdit, Shaper 2D, and 3D Shaper).

Figure 2.16: Various 2D labeled rules for standard shape grammar and the derivations for the rules

Figure 2.17: Various 3D labeled rules for standard shape grammar and the results of the rules
Tapia (1999) developed GEdit, two-dimensional shape grammar interpreter. The system provides an interface that allows users to make or control the rules for spatial layout visually of standard shape grammars [Fig. 2.18]. A designer arranges shapes and defines the rules in the graphic window. The result of the rules is shown in another window.

Mcgill (2000) also developed Shaper 2D as an interpreter for standard shape grammar with visual interface. However, Shaper 2D provide interface that allow a designer to control two rules at the same time and to look at the result in the same window without changing the program [Fig. 2.19]. It helps the designer to test many alternative rules for his or her design in a short time. She tested Shaper2D in a practical studio as a tool for learning shape grammars and applying shape grammars in a design process. Figure 2.20 illustrates the process of applying the generated result to the design process. A designer generated 2D shape configurations in Shaper 2D [Fig. 2.20a], then the designer placed the result on top of the site drawing in another CAD system (AutoCAD) [Fig. 2.20b]. Finally, the designer developed the 2D shape result to the architectural building plan [Fig. 20c].

Figure 2.18: Screenshot of GEdit Interface
Yufei Wang (1999) developed 3D shaper [Fig. 2.21a] as 3D shape grammar interpreter. It provides a dialogue that enables a designer to create 3D objects and define rules. For example, a designer types numerical parameters in the dialogue for the size, type and labels of shapes as well as the spatial arrangement between shapes. Then the system generates 3D forms and creates 3D open inventor files. The designer sees the 3D result through a certain viewer for the open inventor files [Fig. 2.21b].
2.2.2. Algorithmic Shape or Form Generation

Logo Programming Environment supports a designer to create and manipulate shapes by typing a series of instructions. Papert (1967) created the first version of Logo with the team from Bolt, Beranek and Newman, led by Wallace Feurzeig. Since the Logo Programming Environment is easy to use and learn, it has been developed over the past 28-year. The Logo Programming Environment has been used to design programs for various purposes such as constructive learning, mathematics, language, music, robotics, telecommunications, and science.

Designers can explore formal design idea by simply learning how to make an algorithm for shape generation in Logo Programming Environment. Logo Programming Environment uses a Turtle [Fig. 2.22a], which can be directed by typing commands at the computer. For example, the command forward 50 causes the turtle to move forward in a straight line 50 "turtle steps". Right 90 rotates the turtle 90 degrees clockwise while leaving it in the same place. Then forward 50 causes it to go forward 50 steps in the new direction. The turtle generates rectangle shape following the sequential commands in the Logo programming environment [Fig. 2.22b]. The square can be used as a part of another instruction in Logo's vocabulary. Finally, repeating the commands, which rotate 10 degree the rectangle shape like a turtle, can generate the flower-like shape [Fig. 2.22c].
This sequence of commands shows one possible way of generating shape configurations with simple operations. This simple process of Logo Programming Environment is used for many various systems related to various disciplines. So, we can see how simple operations can be applied and advanced to the practical usage.

```
forward 50  right 90  forward 50  right 90
forward 50  right 90  forward 50  right 90
```

(a) the initial turtle (b) to generate “rectangle” shape with turtle commands  (c) to generate “flower” shape by manipulating the rectangle

*Figure 2.22. The turtle commands to generate shape configuration in Logo Programming Environment*

Maeda (1999) developed two-dimensional generative system called Design by Number [Fig. 2.23]. He implemented this system as a tool for teaching the idea of computational design to designers and artists. Design by Number introduced the basic ideas of computer programming within the context of drawing. For example, visual elements such as dot, line, and field are combined with the computational ideas of variables and conditional statements to generate images. Figure 2.23 shows the interface of the system. The interface allows a designer to write simple codes and run the codes and visualize it on the computer screen. Figure 2.24 shows the various results produced using different user-programmed algorithms.
Gross (2001) developed FormWriter, a simple and powerful generative system for generating three-dimensional geometry. With only a few lines of code, a designer generates interesting three-dimensional graphics immediately. He employed a little programming language, which enables architects to explore architectural form by specific simple programming.

The interface mainly consists of editor window for writing code and 3D space with browsing controls [Fig. 2.25]. A designer writes simple commands on the editor window. It positions geometry (triangle, cone, box, sphere, cylinder) in a 3D space. The designer explores the generated forms through the 3D space with browsing controls. Gross used FormWriter in a course on “Computational Geometry in Islamic Architecture.” Students learned how to generate
Islamic Architectural forms using the system. The students programmed codes and generated interesting architectural forms [Fig. 2.26] even though they have no programming knowledge.

Figure 2.25: Snapshot of FormWriter Interface

Figure 2.26: Models of historic Islamic structures by students using the FormWriter

The Processing is a programming environment to generate interactive 2D and 3D images. The idea is to support easy-to-use and simplified codes, which users can understand the concept of programming about creating images and interacting with the images. Using the Processing, a designer generates interesting graphic images as well as learn how to program. Processing supports simplified codes which can run through general Java applets. Figure 2.27 shows the working environment. A designer writes a program in the Text editor window and sees the visual result through the Java applet window.
2.4. DISCUSSION

This chapter introduced some approaches for analysis of style and generative systems for 2D drawing and 3D forms. These two approaches are related to two main studies of this thesis. First the thesis defines a design style particularly in certain drawings with shapes and a set of operations. Second, it proposes a generative system that uses the operations defined in the first study. This section summarizes this chapter and discusses about this thesis.

2.4.1. Analysis of style

The first section described several studies that analyze and describe a design style using shape grammars. They defined a certain design style with a set of replacement rules of shape grammars and generated the original drawings or new drawings in the same style. Shape grammars are used widely. For example, ice-ray grammar used for simple pattern generations; Palladian villas grammar used for complicate architectural plan generations; and Prairie-house grammar used for 3D architectural forms. Shape grammars are also used to analyze a painting style. These approaches show how simple substitution rules of shape grammar can be utilized for analysis of style.
Not using shape grammars, Kirsch used a simple algorithm that modifies a shape to define Miro’s style and Hersey and Freedman’s PlanMaker program used a split system that divides geometry shapes horizontally and vertically following pre-defined split ration. Using the simple split system, PlanMaker allows a designer to generate the original Palladian villa plans and new possible plans in the same style. These approaches demonstrate simple operations or algorithms can be developed to analyze a design style and generate new design in the same style.

Similar with Hersey and Freedman’s approach, this thesis proposes an application process that consists of a set of geometric operations (translation, rotation, and scaling) for a design style. This application process attaches one or several shapes to another shape. Using this process, this thesis demonstrates the generation of Peter Eisenman’s style drawing. Moreover, the application process is further developed as a general method that supports designers to generate shape configurations.

### 2.4.2. Generative Applications

In addition to the previous style analysis applications, this chapter also introduces several generative systems for 2D and 3D drawings. First, it introduces three recently developed shape grammar interpreters that use standard shape grammar. They developed interfaces that allow a designer to control visually the rules of shape grammar. The main purpose of these systems is to support users to learn the shape grammar rules and generate new design in practical design process. Second, we explain generative systems that allow designers to program simple codes to manipulate shapes or forms.

Likewise, this thesis proposes a generative system that uses a simple generation process. As explained previously, this study starts from defining a certain style with a set of operations. From this, the thesis generalizes the generation process and uses it to develop a generative system. This developing process is similar with the shape grammar interpreters that use the shape grammar rules. It is important to provide an interface that allows designers to control the generation process and proceed to new drawings.
In the following chapters, the thesis explains these two approaches. First it describes an analysis application for Peter Eisenman Style, ArchiDNA. ArchiDNA shows the application process to generate Eisenman-like drawings with a set of operations. Next, the thesis introduces ArchiDNA++, a generative system. ArchiDNA++ provides an interface that allow a designer to use the application process by controlling the operations.
Chapter Three
An Eisenman Design Machine

3. ArchiDNA

This chapter describes ArchiDNA, a system for generating designs in the style of Peter Eisenman. It describes Eisenman’s style as a set of operations and demonstrates the use of the operations to generate shape configurations in Eisenman’s style.

ArchiDNA is based on two assumptions. First, a certain design style can be programmed into a set of operations. Second, those operations can be used to generate design works in the same style following a certain process. The prototype ArchiDNA program described in this chapter was built to explore how a designer generates shape configurations, which illustrate a certain design style following a certain process with a set of operations.

This chapter first explains a certain design style with its 2D and 3D drawings. It describes 2D shape configurations with 2D operations and the process of converting 2D configurations into 3D form. Finally it summarizes the generation process of ArchiDNA and discusses the limitations of ArchiDNA.

3.1. EISENMAN’S BIOCENTRUM BUILDING DESIGN

Peter Eisenman is a famous contemporary American architect. He usually creates drawings to find formal design idea from his design concept (Eisenman, 1996). The starting point of our ArchiDNA project was an analysis of Peter Eisenman’s drawings of the design of his Biocentrum building plan in Frankfurt, Germany [Fig. 3.1a]. In this project, Eisenman used drawing both to search for a form and idea, and also to explain how the form and idea can be manipulated as a motif and style (Eisenman, 1996).
Eisenman generated the building form by manipulating shapes that represent the four elements of DNA structure: Adenine (A), Guanine (G), Cyanine (C), and Thymine (T) [Fig. 3.1b]. Four distinct shapes are commonly used to represent these amino acids: arch (A), ribbon (G), pentagon (C) and wedge (T) [Fig. 3.1c], and Eisenman used these as the building blocks for his Biocentrum design. We chose this work because Eisenman’s drawings express geometry that is describable and measurable. We hypothesized that his formal idea could be represented as computer algorithms that manipulate the design components.

Figure 3.1: (a) Biocentrum (Eisenman, 1996) (b) Diagram of DNA showing Amino Acids (c) Four distinct shapes in Amino Acids

Eisenman’s 2D and 3D drawings illustrate how the form of the building came from abstract representations of DNA structure. In the 2D drawing [Fig. 3.2a], four shapes represent the four elements of DNA structure (A-T-C-G), were translated and transformed to compose the final drawing. In the 3D drawing, the 2D drawings were extruded into a 3D form [Fig. 3.2b].
3.2. 2-D OPERATIONS

Analyzing Eisenman’s 2D drawing, we find the primitive operations of the 2D shape configurations. A set of shapes attaches to each other by being translated, rotated, and scaled in Eisenman’s 2D drawing. This section first explains how the primitive operations attach one shape to another. Then it describes the applying process that generates Eisenman-like drawing.

3.2.1. Primitive Operations in ArchiDNA

We attach a shape to another shape following three main operations: translation, rotation, scale. The three operations are basic ways to change a shape: translation (moving it somewhere else), rotation (turning it around) and scaling (making it bigger or smaller). They are best understood graphically first. Figure 3.3 illustrates the three operations (move, rotate, and scale) of rectangle shape to a certain point in 2D Cartesian coordinate space. The three values for the operations are: (10, 10) for location, 45 degree for angle, and 50% for scale. The rectangle is first translated from (0, 0) to (10, 10) [Fig. 3.3a], rotated from 0 to 45 [Fig. 3.3b] and finally scaled with 50% [Fig. 3.3c]. Scaling a shape simply means making it bigger or smaller. A “scale factor” specifies how much bigger or smaller. For example to double the size of a shape we use a scale factor of 200%, to half the size of an object we use a scale factor of 50%.
Figure 3.3: Primitive Shape Operations

Figure 3.4 illustrates the process of applying a rectangle (square) to a line. Given a certain line in the 2D space, we can obtain the three values (translation, rotation, and scale) by getting angle and scale from the 2D line. Simply defining the line, we get the rectangle attach the line [Fig. 3.4]. The line has a location, direction, and a size. So we use the information to move the square to the location, rotate the square to be parallel to the line, and finally scale the square to fit the size of line.

Figure 3.4: A square is attached to a line considering its location, angle, and size

In the same way, if we use a square instead of a line, we can define four squares following the operations (translation, rotation, and scaling). Figure 3.5 illustrates how the rectangle attaches to the four edges of the first square.
3.2.2. Application Process: applier-shape and base-shape

The previous section explained how a shape is translated, rotated, and scaled with another shape in 2D space. ArchiDNA defines the application process as a way of generating shape configurations. It decides how one or more shapes attach to another shape through a sequence of (1) translation [Fig. 3.3a], (2) rotation [Fig. 3.3b], (3) scaling [Fig. 3.3c], and (4) repetition of the three operations [Fig. 3.5d].

In this application process, a shape object may either play the role of an applier-shape (object to be transformed and generated) or a base-shape (a fixed shape object to be attached to by applier-shape objects). Each applier-shape object has an “anchor-edge” that will be used to match and attach to the edges of a base-shape object. The edges of a base-shape, on the other hand, are labeled attachable or un-attachable, indicating whether an applier-shape can be added. Any shape can serve as a base-shape object to generate shape configurations.

Figure 3.5: Four Shape Operations with applier-shape to base-shape
If there is more than one applier-shape object, they attach to a single base-shape object in sequence. For example, Figure 3.6 illustrates the four different shapes (A1-A2-A3-A4) applied to the base-shape B. It starts with the shape A1 and attaches other shapes in sequence, counterclockwise.

Figure 3.6: Four shapes (A1-A2-A3-A4) applied sequentially to the edges of the base-shape B. Note that shape A1 occurs twice because the application process repeats at the beginning once the sequence is exhausted.

3.3. 2-D GENERATION

Following the application process, ArchiDNA generates 2D shape configurations similar to Eisenman’s plans for the Biocentrum building. To demonstrate the process in ArchiDNA, we start with eight shapes (2 copies of each of the four DNA shape elements) [Fig. 3.7].
Figure 3.7: Initial shape in ArchiDNA Interface

The first step is to apply the shape G (ribbon) as applier-shape to the shape C (pentagon) [Fig. 3.8a]. The designer presses the “A” icon for setting applier-shape in the tool palette and selects the G shape. Then to select the C as base-shape the designer presses “B” icon, the designer selects the C shape (pentagon). Now that the designer has provided ArchiDNA a base-shape object and an applier-shape object, ArchiDNA goes to work. ArchiDNA copies and attaches the applier G shapes (ribbon) to every edge of the base C shape (pentagon) [Fig. 3.8a]. The G shapes are attached at the short edge of the ribbon G shape because this is that shape’s anchor-edge.

Figure 3.8b demonstrates this process again with a base-shape A (arch), which has eight edges including five short line segments that approximate the curve. Eight applier G shapes are generated and attached to the eight edges of the base-shape A [Fig. 3.8b]. This example makes clear that ArchiDNA scales the applier shape so that its edge dimension matches that of the base-shape, where it is attached.
Figure 3.8: (a) Application the applier-shape G (ribbon) to the base-shape C (pentagon) (b) Application the applier-shape G to the base-shape A (arch)

We repeatedly click other shapes as base-shape, so that the shape G (ribbon) is applied to other base-shape objects. In this way, we generate interesting configurations quickly. Figure 3.9 illustrates the final drawing looks like Eisenman’s 2D drawing. The shape G is moved, rotated, and scaled based on the edges of the various base shapes selected by the user. This example shows one applier-shape G applied to other shapes. Next time (Fig. 3.10), we use four applier-shape objects to generate an Eisenman-like drawing.

Figure 3.9: ArchiDNA Second Shape Generation
In the same way, we define a series of applier-shapes with the four shapes (A-G-C-T) and select a base-shape repeatedly to generate shape configurations. Figure 3.10 shows four applier-shape objects matched and attached to base-shape objects. Repeatedly selecting base-shape objects, Designer can generate Eisenman-like drawings quickly [Fig. 3.10].

![Figure 3.10: 2D Eisenman-like drawing generated in ArchiDNA](image)

### 3.4 3-D OPERATIONS

Our Eisenman-like drawing can be converted to three dimensions by automatically assigning heights. Each of the four shapes is extruded to a certain height, a function of its area. From Eisenman’s 3D model image [Fig. 3.2b], it appears that a threshold controls the shape extrusion. If the area is larger than the threshold, the height of the shape is assigned a negative value, so that the 3D object extrudes downwards from the ground of the building. Otherwise, the shape will compose a building mass projecting upward. Figure 3.11 shows that the small shape A1 (pentagon) is extruded upwards whereas the larger shape A2 (pentagon) is extruded downwards because its area is larger than the threshold [Fig. 3.11b&3.11c]. On the other hand, the height of shape B (ribbon) is fixed with a user-defined value.
3.5 3-D GENERATION

Following these 3D extrusion operations, ArchiDNA automatically generates 3D objects by extruding the 2D drawings and exporting a 3D VRML file. Figure 3.12 shows a 3D Eisenman-like drawing generated in ArchiDNA. The eight shapes (two pairs of A-T-G-C shapes) in the center are extruded by a certain user-defined height. Other shapes were extruded to a height that is a function of their area.
3.6 SUMMARY

We generated 2D and 3D Eisenman-like drawings using a set of geometric operations of translating, rotating, and scaling applier-shapes and attaching them to the edges of a base-shape. Figure 3.13 shows the four steps of ArchiDNA. Designers define one or more applier-shapes, and then select a base-shape. ArchiDNA generates 2D shape configuration and 3D form. Following this simple process, a designer can generate Eisenman-like drawings in a very short time. Even a beginner can learn and use ArchiDNA.

3.7 DISCUSSION

We developed ArchiDNA to generate Eisenman-like configurations according to a simple system of operations. The designer defines applier-shape objects and selects base-shape objects. The duplicates of applier-shape objects attach to the edges of the selected base-shape object. ArchiDNA shows a simple and interesting process to generate Eisenman’s drawings. It was easy to use and learn. We quickly generate a complicate shape configurations and 3D forms in
the Eisenman’s style. However, the drawings ArchiDNA generates are confined to Eisenman-like drawings. From this, some limitations of the current ArchiDNA are described like below:

- ArchiDNA operates with four built-in shapes (A-T-G-C).
- A designer cannot control the shape operations (rotation and scaling).
- A designer cannot change the shape attributes (attachable-edge of base-shape object and anchor-edge of applier-shape).
- ArchiDNA interface does not support the designer to control the sequence of applier shapes.

If the study of ArchiDNA stops after generating Eisenman-like drawing, ArchiDNA will be used to learn or understand Eisenman’s drawing without being further utilized in design process. It seems to be possible to generalize the generation process as a method to generate 2D shape configurations. A system can support a designer as a generation tool for shape configurations. If a system allows a designer to control the process, the designer can use the system to generate configurations and 3D forms for his or her design. How can we use the application process to generate other drawings? The thesis proposes the extension of ArchiDNA, called ArchiDNA++, that makes the generation process serve the general shape configurations. Next section describes ArchiDNA++ system which supports a designer to control the geometric operations and creates his or her own shapes for applier-shape and base-shape. It also explains how a designer can utilize the ArchiDNA++ to produce shape configurations.
Chapter Four

ArchiDNA++
An End-User Programming Environment For Shape Application

4. ArchiDNA ++

ArchiDNA++ is an extension of ArchiDNA as a generative system for shape configurations. The development of ArchiDNA++ focuses on how to use the application process with a set of operations used for Eisenman-like drawing in Chapter 3. Thus, the first goal is to incorporate the Application process to generate other stylistic shape configurations. It can help a designer to generate shape configurations for his or her design work.

In the previous section, We noted some limitations of ArchiDNA to address these limitations. ArchiDNA++ provides an interface that allows users to proceed to their own design project controlling the operations. ArchiDNA++ has the following features to support designers to develop shape configurations:

1. A designer creates his or her own shapes as applier-shape and base-shape.
2. A designer manages applier-shape list for the order of applier-shapes.
3. A designer changes shape operations and attributes for modifying shape configurations.
4. A designer generates shape configurations by drawing a base-shape as well as selecting a base-shape.
5. A designer controls 3D conversion by setting height or threshold.

The chapter is divided into four sections: Section 4.1. describes the work environment of ArchiDNA++. Section 4.2. explains overly the overview of ArchiDNA++ operations. Section 4.3. describes shape preparation including shape creation and attribute control for. Section 4.4. explains shape configurations with the prepared shapes as applier-shape and base-shape. Section
4.5 discusses the conversion from 2-D to 3-D. Finally Section 4.6 introduces an example to generate a sunflower shape explaining ArchiDNA++.

4.1. ARCHIDNA++ ENVIRONMENT

The working environment is an important feature of ArchiDNA++. When generating drawings in ArchiDNA, the generation process was hidden from the designer; the designers could not control the process. However, most designers want to use their own shapes and control the rules while generating shape configurations because design work represents a designer’s personal thinking. ArchiDNA++ attempts to make the configuring process as transparent as possible through the interface providing an efficient way to create shapes and control the generation process.

The main purpose of developing the working environment is to allow a designer to create applier-shape and base-shape for his or her own design. For this purpose, the system must provide an interface to support the designer in generating shapes and to program the operation of shape configuration.

The working environment of ArchiDNA++ has three main parts [Fig. 4.1]: at top a tool palette, at left a window consisting of a drawing panel and a configuration panel with a view palette; at right a control window. The control window has four palettes: a drawing palette [Fig. 4.2a], an attribute palette [Fig. 4.2b], a list palette [Fig. 4.2c], and a 3D palette [Fig. 4.2d].

4.1.1. Tool Palette

The tool palette is at the top of the screen and consists of eight buttons. Pressing each button at the first row displays its corresponding panel in the control window at the right. Four buttons on the second row are used to export 3D VRML files, saving and opening an ArchiDNA++ file, and showing version information about this system.

4.1.2. Drawing Window
The Drawing Area is composed of three panels divided horizontally: the configuration panel, drawing panel and view palette. The configuration panel (upper) is for generating shape configurations, and the drawing panel (lower) is for drawing shapes for applier-shape and base-shape. Users can change the size of each panel by moving the resize button between the two panels button up and down. This separated drawing area helps users to draw shapes and set applier- and base-shapes. The view palette at the bottom allows users to zoom in or out, turn grid and grid snapping on and off for the configuration panel and drawing panel.

### 4.1.3. Four Control Palettes

Once the designer creates shapes in the drawing panel using the drawing palette, the designer controls shape attributes with the attribute palette and defines applier-shape(s) using the list palette before generating shape configurations. ArchiDNA++ has four control palettes displayed in the control window: drawing palette, attribute palette, list palette, and 3D palette [Fig. 4.2]. The designer switches from one to another by simply pressing the four relevant buttons in the tool palette.

![Figure 4.1: Snapshot of ArchiDNA++ Interface](image-url)
4.2. ARCHIDNA++ IN USE

A designer uses ArchiDNA++ in three phases: Shape Preparation, Shape Configuration, and 3D Conversion. Figure 4.3 illustrates the three phases with five new steps (gray boxes) in addition to the steps described in the previous section [Fig. 3.13]. It supports an easy and powerful process not only to generate Eisenman-like drawings but also to describe and generate drawings in other styles. We next describe this process in detail.
4.3. SHAPE PREPARATION PHASE

The Shape Preparation Phase prepares shapes to be used as applier-shape and base-shapes. If a designer already has previously prepared shapes, this phase can be skipped. This phase has two sub-processes: Create Shapes and Control Shapes. Once the designer creates a shape, it is not necessary to change it for shape configuration.

4.3.1. Shape Creation

The designer can create various shapes in the drawing panel [Fig. 4.5 & bottom of Fig. 4.1]. For this, ArchiDNA++ provides tools for drawing (line, poly-line, rectangle, sketch, and text) and editing (select, move, rotate, and delete) in the drawing palette [Fig. 4.2a] like other CAD software. For example, to draw a rectangle shape, the designer first selects a button for “draw rectangle” in the drawing palette, and clicks a mouse button in the drawing panel. A rectangle is rubber banded as the designer moves the mouse. When the designer clicks on another point, a rectangle is created and displayed in the drawing panel.
The sketch tool lets the designer draw continuous lines by clicking and dragging, rather than a click-click process, as with line, rectangle, and poly-line. The designer selects the button for “draw sketch”, click to begin, and then drag the mouse freely to draw some shape. Then the designer releases the mouse to stop drawing. The sketch line drawn consists of little line segments, which are generated by the program automatically whenever the mouse travels a certain time. Then these lines are used to generate the sketch line. The distance can be varied by changing the drawing speed.

Figure 4.5 also shows default shape attributes of a rectangle shape (name and area of the shape; direction, angle, and scale-factor of each edge). A user can control the Application process by changing these attributes. The next section explains controlling the attributes in detail.

![Figure 4.5: Various Created Shapes; a rectangle (□) in the middle of an edge marks an anchor-edge.](image)

### 4.3.2. Shape Attribute Control

In Figure 3.5, we generated a certain shape configuration applying an applier-shape object (ribbon) to a base-shape object (pentagon). A designer can generate different shape configurations simply by changing shape attributes: anchor-edge, attachable-edge, operation values (angle and scale), and direction. For this, the system provides an attribute palette [Fig. 4.2b].

Figure 4.6 illustrates how the designer changes the anchor-edge from the top-side edge (default) to the right-side edge in an applier-shape object. The system generates a different configuration, matching and attaching the right-side edge to all edges of a base-shape object.
The designer can change attachable-edges to un-attachable edges. Figure 4.7 shows when two edges of the base-shape object are made un-attachable, Those two edges of the base-shape object do not have any instance of the applier-shape object.

The designer can also control shape operations (rotation and scaling) by setting operation values (angle and scale-factor) to an edge of a base-shape object. By default, the applier-shape object matches and attaches to an edge of a base-shape object [Fig. 3.5]. Figure 4.8 shows a different result of fixing the base-shape’s three edges with a certain angle (90°) and a scale factor (50%). The three applier-shape objects were translated to each edge of the base-shape object and rotated and scaled with user-defined values (not matching the base-shape object).
Figure 4.8: Fix three edges with angle (90°) and scale-factor (50%) and its result shows three applier-shape objects are not matched to the base-shape object.

In addition to these three attributes, a designer can reverse the direction of a base-shape object from counter-clockwise (default) to clockwise. Figure 4.9a shows how four applier-shape objects (A1-A2-A3-A4) attach to the default base-shape object in sequence following the direction, counter-clockwise [also see Fig. 3.6]. If the designer changes the direction to clockwise, the four applier-shape objects attach in sequence, clockwise, generating a different shape configuration [Fig. 4.9b].

Figure 4.9: (a) Counter-clockwise base-shape object (b) Clockwise base-shape object with four applier-shape objects (A1-A2-A3-A4)
4.4. SHAPE CONFIGURATION PHASE

The Shape Configuration Phase generates shape configurations with applier-shapes and base-shapes. In the configuration panel, designers proceed to the “application-process” using the shapes drawn in the drawing panel.

4.4.1. Define applier-shape(s)

The list palette consists of two sub-lists: an applier-shape list (upper) and an all-shape list (lower) [Fig. 4.2c]. A designer can see the sequential list of applier-shapes and all shapes. The designer defines applier-shape list using drag and drop from the all-shape list to the applier-shape list.

4.4.2. Select base-shape or Draw base-shape

Designers can generate new shapes by choosing or drawing a base-shape object in the configuration panel. When drawing a base-shape, designers use the drawing palette in the same way, as one would do in creating a shape in the drawing panel.

Figure 4.10 shows various shape configurations generated in ArchiDNA++. Figure 4.10a illustrates result of drawing a curved line as a base-shape. As small-segmented lines approximate the curve line, the instances of the applier-shape (ribbon) are attached to the curve following the segmented lines. Figure 4.10b shows an interesting result when a ribbon-like shape is used as applier-shape and base-shape at the same time. Figure 4.10c shows what happens when a single human figure as applier-shape is applied to a rectangle base-shape. It generates a pattern reminiscent of the work of M.C. Escher.
4.5 3-D CONVERSION

ArchiDNA++ converts 2D shape configurations into 3D forms by assigning heights to each object according to its shape. The system assigns the height automatically using the area value of a closed-shape or the boundary length of an open-shape. A designer can also fix a height of a shape by defining a certain value through the 3D palette in the control window [Fig. 4.2d].

4.5.1. Set a Threshold

As explained previously [Fig. 3.11], 3D conversion in ArchiDNA operates on a certain threshold to decide either the shape is extruded upward or downward. A designer can specify a threshold value by selecting a shape of which area (closed shape) or length (open-shape) to be used as a threshold.

4.5.2. Set 3D Mode
Once a threshold is set, the designer can choose a mode (block or wall) to generate a 3D-object in either block-type [Fig. 4.11a] or wall-type [Fig. 4.11b]. Block mode creates a mass model extruding the 2D shape configurations as we generated 3D Eisenman-like drawing [Fig. 3.2]. Wall mode converts a 2D line to a 3D wall object with a preset thickness that a designer can define in the 3D palette [Fig. 4.2d]. Figure 34 shows the difference between the block mode and the wall mode. The designer can display the converted 3D VRML model with VRML viewer (e.g. Cortorna, Cosmo, etc) on a web-browser [Fig. 4.12a]. He or she can also export the VRML file into other 2D or 3D formats for further manipulation such as rendering or animation [Fig. 4.12b].

Figure 4.11: 3D Mode: block-mode and wall-mode
(a) 3D model generated in block-mode (b) 3D model generated in wall-mode.
4.6. EXAMPLE: GENERATION OF SUNFLOWER SHAPE

This section shows how a designer uses ArchiDNA++ to generate a sunflower shape [Fig. 4.14]. The scenario presented here helps to clarify and demonstrate how to use the application process using ArchiDNA++.

The scenario is divided into two steps: shape preparation and shape configuration. The preparation creates three shapes (triangle, hexagon, and ribbon) [Fig. 4.13] and controls those shapes in terms of their use as applier-shape and base-shape. The configuration phase uses those prepared shapes to generate shape configuration. There are two steps. The first step generates sunflower-like configurations by attaching the ribbon applier-shape to the edges of the hexagon base-shape [Fig. 4.15] and the second step elaborates the first shape configuration by applying the triangle applier-shape to all ribbon shapes generated in the previous step [Fig. 4.16].

4.6.1. Shape Preparation

1) Create three shapes (hexagon, triangle, and ribbon) on the drawing panel.

The designer presses the Shape Drawing Control button, and the drawing palette appears in the right control area, which has drawing tools such as line, rectangle, poly-line, sketch, and etc. By choosing the “draw poly-line” button, the designer can create three shapes (hexagon, triangle, and ribbon-like shape) in the drawing panel.

First, the designer clicks on a point, and then clicks on another line. After each click, a new line is rubber banded that is anchored on the previous point. The designer double-clicks on the point he wants to finish. A polygon shape is created and displayed. The double click of the mouse signals completion of the drawing process and creates a 2D shape. Positioning the mouse on the first point and double clicking closes a shape. Following these steps, the designer creates a hexagon, triangle, and ribbon shape on the drawing panel [Fig. 4.13].
2) Change the anchor-edge of the triangle

To attach the longest edge of the triangle to a ribbon, the designer needs to change the anchor-edge. When creating a triangle, the default anchor-edge was right diagonal edge [Fig. 4.13]. The designer can change the anchor-edge to the bottom edge by using the “Set anchor-edge” button in the attribute palette. After pressing the button, the designer clicks the bottom edge and then the red-rectangle mark will be changed to the bottom edge [Fig. 4.14b].

3) Change the attachable edge of the ribbon-like shape

In the next step, the designer chooses a triangle for the applier-shape, and applies it to each base-shape (ribbon). We will attach the triangle to the two side edges (right and left) of the ribbon shape, so only two side edges should be defined as “attachable” and other edges as “un-attachable”. To change an edge to un-attachable, after pressing the button for “set attachable” in the attribute palette, a designer selects an attachable edge. The set-attachable button toggles in attachable mode of an edge. Figure 14a shows that the two edges are active sites where other shapes may be attached.
4.6.2. Shape Configuration

1) Place the hexagon as an initial base-shape to Shape Configuration Panel

The shape configuration step starts from locating the hexagon shape to the configuration panel. We will attach the ribbon shape to the hexagon. First the hexagon should be moved to the configuration panel as a base-shape object. Once the designer creates the three shapes (ribbon, hexagon, and triangle), they are automatically added to the all-shape list with a default name (Shape 1, Shape 2, and Shape 3). When selecting the “Shape 2” in the shape list, designers can see the hexagon shape is highlighted with a red color. A designer moves the hexagon by dragging it to the configuration panel and clicks on the configuration panel.

2) Set the ribbon shape to applier-shape list

The designer also must set an applier-shape object. We will use the ribbon shape as the applier-shape object. Similar to the previous example, the designer selects the “Shape 1” that represents the ribbon in the all-shape list, and then drags the item to the applier-shape list, adding the ribbon to the applier-shape list.

3) Generate Sunflower-like shape in the configuration panel

Now a designer can generate the first shape configuration with the applier-shape object (ribbon) and the base-shape object (hexagon). First the designer presses the “select base” button in the drawing palette. The designer selects a hexagon base-shape in the configuration panel. Figure 4.15 shows the resulting final shape in which ribbon shapes are attached to all edges of the hexagon base-shape object.
Figure 4.15: *First shape configuration for sunflower-like shape attaching ribbons to a pentagon*

4) **Set the triangle to applier-shape list**

This time we use the triangle as an applier-shape and attach to the already generated ribbon shapes. First we must change the applier-shape list. The designer selects the Shape 1 that represents the ribbon we used in the previous configuration, and then moves it to the all shape list. This deletes it from the applier-shape list. Then, in a similar way, the designer moves the triangle from the shape list to the applier-shape list.

5) **Select the already-generated ribbons as a base-shape object**

The final step is to generate the second configurations that elaborate the first configuration with a triangle. When a designer select a ribbon, the triangle applier-shape is attached to the two side edges of the ribbon. The triangles are not attached to the two diagonal edges at top of the ribbon because they were labeled un-attachable. In this way, the designer selects all ribbons as a base-shape object and makes the elaborated sunflower [Fig. 4.16].

Figure 4.16: *Second shape configuration for sunflower-like shape attaching a triangle to the ribbon base-shape objects*
5. Implementation

This chapter explains how ArchiDNA++ is implemented. The first ArchiDNA system is a Java Applet. It is implemented using JDK (Java Development Kit) 1.0. The system runs on any web-browser that supports JDK 1.0. This ArchiDNA++ software used some Java classes of JDK 1.4 for implementing GUI (Graphic User Interface). A designer must install JRE (Java Runtime Environment) 1.4 in their local machine.

The implementation of ArchiDNA is divided into 3 parts:

- Shape Data Structure: to represent a shape as applier-shape and base-shape
- Shape Configuration: to generate shape configurations
- Exporting VRML models: to convert 2D continuations into 3D forms

5.1. SYSTEM ARCHITECTURE

ArchiDNA++ consists of three main parts: data structure, configuration system and 3D converter. Figure 5.1 illustrates the ArchiDNA++ system architecture.
Configuration system generates shape configurations using the shapes created in the drawing panel. It gets shape information from the data structure. The generated shapes in the configuration system are stored again in the data structure. Data Structure: data structure hierarchically manages the sub-shapes (points and lines) and represents the shapes on the computer screen. When designers draw shapes in the drawing panel, this drawing information is sent to this data structure. 3D converter translates the 2D configurations into a 3D VRML format. It is connected to the data structure to get the 2D shape information.

5.2. SHAPE DATA STRUCTURE

ArchiDNA is a 2D vector-based system. A shape in the system consists of sub-shapes (point and edge). The application process uses the sub-shapes when attaching an applier-shape object to a base-shape object. For this, ArchiDNA has a simple data structure to represent various 2D shapes.

Figure 5.2 shows the process to construct a ribbon-like shape with sub-shapes (point and edge). First, the system creates five points by specifying five 2D coordinates [Fig. 5.2a]. Then, the five points are connected with five edges [Fig. 5.2b]. One of the edges is marked with a star (*) as...
anchor-edge. All edges are marked with a plus (+) for attachable-edge or a negative (-) for un-attachable-edge, and operation values (angle and scale).

Figure 5.2: Representation of a ribbon-like shape with points and edges
(a) five points with 2D coordinates (b) five edges with anchor-edge mark [*], attachable-edge mark [+], un-attachable-edge mark [-], and operation values [angle and scale] (c) the final display on the screen.

Figure 5.3 illustrates the hierarchical and object-oriented data structure for a ribbon-like shape. A shape object references five edge objects and has an anchor-edge. Each edge object references two point objects as a start-point and an end-point, and has an attachability (+ or -) and an angle and a scale for shape operation. A point object has a X-coordinate and a Y-coordinate.
5.3. SHAPE CONFIGURATION

As explained before, ArchiDNA attaches one or several applier-shapes to a base-shape in order to generate a shape configuration. Figure 5.4 shows how the system generates shape configurations by referencing shapes. The system has a shape list to store all created shape objects. The applier-shape list is constructed with a shape ID. Therefore when users select an applier-shape, the ID of the selected shape is added to the applier-shape list. Using the IDs and their sequence in this list, instances of shape objects are linked to the edges of the base-shape. The Shape Operations module transforms the instances to match and attach to the edges. For this, the Shape Operation module sets a location for translation and calculates an angle for rotation and a scale-factor for scaling, using the points of the base-shape.
### 5.4. EXPORTING VRML MODELS

ArchiDNA converts 2D shape configurations to 3D objects in VRML format. VRML is widely used to visualize 3D models on the Internet. Most 3D modelers can import this format. Figure 5.5 describes the simple process that converts a 2D ribbon shape to 3D face objects of VRML. First using the x and y coordinates of the ribbon shape, the system constructs two 3D polygons. Each polygon is constructed by adding a Z-coordinate to the 2D coordinates of the 2D ribbon shape for height. The first polygon assigns 0 to Z-coordinate and another assigns the shape’s area (closed shape) or length (open shape). The system sets an ID number to each 3D coordinate in sequence. Then it generates seven 3D VRML face objects by linking the relevant IDs. In case of an open shape, only side faces are constructed without a top and bottom face. Through these simple steps, the ArchiDNA software converts 2D shape configurations to a 3D VRML file format.
Figure 5.5: Converting a 2D shape (ribbon) to 3D VRML objects.
Chapter Six

6. Conclusion and Future Work

6.1 CONCLUSION

This thesis has developed a generative system, ArchiDNA to generate a particular Eisenman’s drawing. It has also developed the extension of ArchiDNA, called ArchiDNA++ to allow a designer to generate general drawings. Developing these two systems, the thesis described two topics: analysis of design style and generative design system. As a result, this study showed how a generative system can be developed from a design style.

ArchiDNA demonstrates that a particular Eisenman’s drawing is generated following simple generation process with a set of operations, called an application process. Application process attaches one or several shapes to another shapes following the operations (translation, rotation, scaling). In ArchiDNA, a designer starts the application process with a set of shapes (represent four DNA components, A-G-C-T). The designers define one or several shapes as applier-shape and select another shape as base-shape. Then the system attaches the applier-shape objects to the edges of the base-shape object. Keep selecting shapes as base-shape, the shapes are configured each other and generate an Eisenman-like drawing. This simple application allows a designer to generate Eisenman-like drawing. ArchiDNA also converts the generated 2D Eisenman-like drawing into 3D forms look like Eisenman’s 3D model for his 2D drawing.

The application process in ArchiDNA is a simple way of describing Peter Eisenman’s design style especially in a certain drawing. Designers learn the generation process and generate drawings in the same style. It is simple and easy to learn and use. Thus even first time user or beginning user easily generates Eisenman-like drawing. However, ArchiDNA uses the application process only to generate Eisenman’s drawing. How can a designer use the powerful generation process for their design? This was the second question of the thesis.
ArchiDNA++ is a generative system for the use of the application process. This system provides an interface to control the generation process. In ArchiDNA++, a designer creates his or her own shapes and controls the shapes for applier-shape and base-shape. Then the designer uses the newly created shapes for the application process. The designer generates his or her own drawings following the same way used for the Eisenman-like drawing. The designer also converts the generated drawings into 3D forms and moreover controls the 3D converting process. It enables the designer to use the application process in the practical design process. Using ArchiDNA++, designers construct both an advanced understanding of the application process and utilize its operations in their design process.

6.2 FUTURE WORK

ArchiDNA is a working system. It efficiently manages generation of shape configurations and its operations. However, there are many more features one could add. This section describes three main directions.

6.2.1. Spatial Configurations

ArchiDNA can become a tool to manipulate and transform spatial arrangements. In sum, ArchiDNA has an object-oriented data structure to represent and operate shapes. It can be expanded to a simple building data model that represents several building components such as space, wall, and column. For example, we can start by substituting shape, edge and point object to space, wall and column object respectively in a simple way. Then we can add opening object (window or door) to the wall. This process could be a starting point to develop ArchiDNA as a system to generate spatial arrangements of a floor plan.

Figure 6.1 shows the possible process to represent a ribbon space with wall, column, window, and opening. Figure 6.2 illustrates for a building data model for the ribbon space. Comparing with the previous data structure for a ribbon shape in ArchiDNA, this data structure has space, wall, column, window, and door object. Figure 6.3 describes the possible 3D form converted
from the 2D ribbon space. The 3D wall objects are extruded with a certain thickness and have an opening like door or window.

Figure 6.1: Representation of a ribbon space

Figure 6.2: Data structure for a ribbon space

Figure 6.3: 3D conversion of a ribbon space
6.2.2. Application to Shape Grammar

ArchiDNA uses the application process to generate shape configurations to generate Eisenman-like drawing and new possible drawings. It has a set of operations (translation, rotation, scaling) that attach one or several shapes to the edges of another shape. The application process can be implemented with replacement rules of shape grammar. We can make a possible scenario that use shape grammar rules for this process. When a designer defines one or several shapes as applier-shape, each applier-shape is defined as a replacement rule of shape grammar. For instance, a single-line shape is positioned on the left side of the arrow and one applier-shape on the right side of the arrow. The applier-shape consists of line shapes, one of which has a labeled line. The labeled line of applier-shape indicates a matching line of applier-shape when replaced with the line. If there is more than one applier-shape, the replacement rules are defined corresponding to the applier-shapes. Once the rules are fixed, the application process can start replacing the lines with applier-shapes. Following the sequence of lines, the rules are applied recursively and a shape configuration is generated.

Figure 6.4 illustrates possible shape grammar rules for the application process of ArchiDNA. Rule 2 shows one single line is replaced with a ribbon shape. The ribbon shape consists of the sequential five lines and one of the lines is labeled with a dot. The number that indicates the sequence is used when applying the rules sequentially. The derivation shows the ribbon shape sequentially replaces the line of the first ribbon shape following the numbered order of the lines. Figure 6.5 shows the modified rules for the case there is more than one applier-shape (ribbon and pentagon). The rule 2-1 replaces one single line with ribbon shape and the rule 2-2 with a pentagon shape. In derivation, these two rules are applied in sequence. Figure 6.6 also shows that the rule 2-1 is applied first and then rule 2-2 is applied to the next line following the sequential order of lines.
6.2.3. 3D Form Configuration

The current application process uses 2D shape. ArchiDNA can be developed to use 3D shapes which consist of face object instead of 2D shapes for the application process. A possible scenario can be described like this. A designer defines one or several 3D applier-shapes then select 3D base-shape. Then the system attaches 3D applier-shape to the faces of the 3D base-shape. Each face has an instance of 3D applier-shape transformed with the three operations (translation, rotation, and scaling). However, the operations of this case should be functioned with three-dimensional coordinates (x, y, and z) in 3D space. This 3D application process enables exploring 3D formal design idea by simple applying 3D shapes to another 3D shape. Figure 6.6 shows one 3D box applier-shape as applier-shape is applied to another 3D box base-shape and generate interesting 3D form configuration.
Figure 6.6: 3D form configuration of applying 3D applier-shape to 3D base-shape
Bibliography


Knight, T.W.: 2000, Shape Grammars in Education and Practice: history and prospects, http://www.mit.edu/~tknight/IJDC/ (online paper), Department of Architecture, MIT.


MIT Logo project, http://el.media.mit.edu/logo-foundation/

Processing, http://proce55ing.net/


