Ain Shams University
Faculty of Engineering
Department of Architecture

FORM GENERATION IN ARCHITECTURE

USING TOOLS BASED ON EVOLUTIONARY AND MATHEMATICAL FUNCTIONS

By

Ahmed Medhat El Iraqi

B.Sc. Architecture 2002- Ain Shams University

A Thesis Submitted in Partial Fulfillment
Of the Requirements of

M.Sc. Degree in Architecture

Supervised by

Prof. Yasser Mansour
Prof. of Architecture
Head of Architecture Department
Faculty of Engineering
AinShams University

A. Prof. Gamal El Kholy
A. Prof. of Architecture
Department of Architecture
Faculty of Engineering
AinShams University

A. Prof. Ashraf Abd El Mohsen
A. Prof. of Architecture
Department of Architecture
Faculty of Engineering
AinShams University

2008

Ain Shams University
Faculty of Engineering
Department of Architecture

FORM GENERATION IN ARCHITECTURE
USING TOOLS BASED ON EVOLUTIONARY AND MATHEMATICAL FUNCTIONS

A Thesis Submitted by
Ahmed Medhat El Iraqi

B.Sc. Architecture 2002- Ain Shams University
In Partial Fulfillment of the Requirements of

M.Sc. Degree in Architecture
Date: / /2008

Jury committee:

Ain Shams University
Faculty of Engineering
Department of Architecture

STATEMENT

This thesis is submitted to Ain Shams University for “The Degree of Master of Science in Architecture”. The work included in this thesis was accomplished by the Author at the Department of Architecture, Faculty of Engineering, Ain Shams University, during the period from 2005 – 2008.
No part of this thesis has been submitted for a degree or a Qualification at any other university or institute.

Name: Ahmed Medhat El Iraqi

Signature:

Date: / /2008

بسم الله الرحمن الرحيم

Dedication

To

My Adorable Mother

My Great Father

My Caring Sister

My Loving Fiancée

My Dear grandfather

My Supporting Family

And all whom I love…

I dedicate my humble research

Ahmed Eliraqi

ACKNOLEDGEMENTS

I would like to thank Allah for his guidance and support and for giving me the strength and patience to complete this thesis. I would like to express my deep and sincere gratitude to my supervisor, Professor Yasser Mansour, Prof. of Architecture and Head of Architecture Department Faculty of Engineering AinShams University, for his constructive comments. His logical way of thinking has been of great value for me. His understanding, encouraging and guidance have provided a good basis for the present thesis. And I would like to warmly thank my supervisor, Assistant Prof. Gamal El Kholy, for his support throughout this work. His concepts had a remarkable influence on my thesis. I am also grateful to Assistant Prof. Ashraf Abd Elmohsen for his understanding and encouraging. Also I would like to thank Dr. Magdy Ibrahim for his assistance in explaining some complicated techniques used in this thesis and providing helpful insight when needed. I would also like to thank and dedicate this work to my family for supporting and encouraging me to compile this work. And I would like to express my deep gratitude and appreciation for my loving and caring fiancée Mona, for her patience and support and for giving me the strength and courage to continue writing this thesis when I didn’t know I could.

FORM GENERATION IN ARCHITECTURE

USING TOOLS BASED ON EVOLUTIONARY AND MATHEMATICAL FUNCTIONS

ABSTRACT

Form finding process is developed with the reciprocity between advances in mathematics and IT. This elaborates new mathematical design tools with higher mathematical and evolutionary basis, using algebraic, topological, trigonometric, chaotic and evolutionary functions. These tools changed the theoretical and mathematical perceptual natures of architectural form, from platonic Euclidean geometry to unpredictable new geometries.

The thesis juxtaposes seven tools, classified into three mathematical generative systems, aiming to point to and identify new methodologies or tools that can generate unpredicted novel forms, to trigger inspiration and empower inventiveness during the form seeking dilemma in the architectural design process. This will be done through describing how mathematics had been shifted from being an organizing tool into a creative tool then a generative medium, and describing the conventional mathematical generative systems and listing its architectural potentials, then explaining how mathematical elements in the world of chaos and random functions and evolutionary process in natural systems can be simulated artificially and interpreted into elements with architectural potentials.

Deducing the characteristics and promises of these generative systems, with listing some applications in architecture and analytically comparing the efficiency of each generative tool in different applications in the architectural design process, will be shown in the discussion. Finally, results and recommendations will be presented at the end of the thesis.

LIST OF CONTENTS

Chapter 2: Conventional Mathematical Generative Systems

Chapter 3: Chaos Based Mathematical Generative Systems

Chapter 4: Evolutionary Based Mathematical Generative Systems

Chapter 5: Discussions, Results and Recommendations

LIST OF FIGURES

Figure		Page no.

Figure **Page no.**

Figure **Page no.**

Figure		Page no.

Figure		Page no.

Figure **Page no.**

LIST OF TABLES

LIST OF TERMS

Algorithmic Generation:

Alteration of Middle English *algorisme,* from Old French & Medieval Latin *algorismus,* from Arabic *al-khuwārizmi,* from *al-Khwārizmī fl* A.D. 825 Islamic mathematician. **a:** a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation. **b:** a step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

Cellular Automata:

Cellular: of, relating to, or consisting of cells, and automata: **a:** a mechanism that is relatively self-operating, **b:** a machine or control mechanism designed to follow automatically a predetermined sequence of operations or respond to encoded instructions, **c:** an individual who acts in a mechanical fashion.

Chaos Mathematics:

Unpredictable behavior of deterministic systems has been called chaos. And chaos mathematics is the branch of mathematics that deals with the nature and the consequences of chaos and chaotic systems**.**

Chaotic Process:

Chaotic processes are not random process; they follow rules, but even very simple rules can produce extreme complexity. This complexity can be expressed as a series of equations or visualized and rendered when the element of time as color is introduced into its interpretation. The mathematics of chaos provides the tools for creating and displaying such phenomenon.

Evolutionary Process:

A process of continuous change from a lower, simpler, or worse to a higher, more complex, or better state. According to the theory of evolution which states that the various types of animals and plants have their origin in other preexisting types and that the distinguishable differences are due to modifications in successive generations.

Expert Systems:
Is computer software that attempts to mimic the reasoning of a human specialist, and based on initial studies in limited design domains, such as office design, kitchen design and layout generation applications.

Fractals:
French *fractale*, from Latin *fractus* broken, uneven. Any of various extremely irregular curves or shapes for which any suitably chosen part is similar in shape to a given larger or smaller part when magnified or reduced to the same size.

Generate:
a: To originate by a vital, chemical, physical, mechanical or mathematical process or to be the cause of (a situation, action, or state), **b:** to define (as a mathematical or linguistic set or structure) by the application of one or more rules or operations; especially: to trace out (as a curve) by a moving point or trace out (as a surface) by a moving curve.

Generative:
Having the power or function of generating, originating, producing, or reproducing.

Generative Systems:
Systems that use a few basic rules to yield extremely varied and unpredictable patterns as well as anticipated ones. Conway's Game of Life is an excellent example of one such system: Cellular automaton. These systems can be found in music, Generative music, in art, Generative art, and, more recently in architecture, Generative design.

Generativity:
A new term describing the state of applying generative systems (generators) in some fields like architectural design, differs from the essence of creativity, which is the ability to produce or create via (intuition).

Genetic Engineering:
The group of applied techniques of genetics and biotechnology used to cut up and join together genetic material and especially DNA from one or more species of organism and to introduce the result into an organism in order to change one or more of its characteristics.

Genetic Algorithms:

Is one of the evolutionary methods, which is able to generate unexpected, novel forms. This technique is based on the idea of survival of the fittest. GAs consists of a population of solutions, called phenotypes, which are represented by their genetic code, or genotypes. The fittest phenotypes are chosen from the population of candidate solutions by the means of fitness function. The corresponding genotypes are used to create new and conceivably better populations of solutions, by crossing over and mutation.

Genotype:

The genetic code script of all or part of the genetic constitution of an individual or group.

Phenotype:

The observable properties of an organism that are produced by the interaction of the genotype and the environment

Parametric Variations:

Is changing an arbitrary constant whose value characterizes a member of a system (as a family of curves), or changing an independent variable used to express the coordinates of an element.

Strange attractors:

Are generated repeating point patterns in two-dimensional space with a coloring algorithm that can produce images of coherent three-dimensional forms. The third dimension is determined by the perception of the viewer coupled with a created intent. The iterated points can also be plotted in real three- dimensional orbit.

Shape grammars:

Are shape rule that defines how an existing shape or part of a shape can be transformed. A shape rule consists of two parts separated by an arrow pointing from left to right. The left part of the arrow depicts a condition in terms of a shape and a marker. The right part of the arrow depicts how the left shape should be transformed and where the marker is positioned. The marker helps to locate and orientate the new shape.

Spirolaterals:

Are geometric entities generated with a mathematical rule based in a computational medium with increasing length of turns and turns repeating themselves under defined angle. They resemble fractals in some chaotic properties.

LIST OF ACRONYMS

ACRONYMS	STANDS FOR
AI	Artificial Intelligence
BASIC	Beginner's All-purpose Symbolic Instruction Code
C and C++	Programming Languages (not acronyms)
CA	Cellular Automata
CAAD	Computer Aided Architecture Design
CAD	Computer Aided Drafting or Design
CCTV	China Central Television
CNC	Computer Numerically Controlled
DNA	Deoxyribo Nucleic Acid
GA	Genetic Algorithm
GDL	Gnu Data Language
GE	Genetic Evolution
IFS	Iterated Function Systems
JAVA	Programming Language (not acronym)
LISP	list processing language
MathCAD	Mathematical Computer Aided Design
MEL	Maya Embedded Language
NURBS	Non-Uniform Rational B-Splines
TCl	Tool Command Language
TK	Tool Kit
OMA	Office for Metropolitan Architecture
VRML	Virtual Reality Modeling Language

INTRODUCTION

INTRODUCTION

Mathematics had always played a significant role in the process of form finding in art and architecture through history. The reliance was on classical mathematics and Euclidean geometry for ages. This reliance continued in the beginning of the 20th century due to the inability to perform complex mathematical calculations to create complex forms and the lack of the tool to visualize these complex forms.

In the last decade of the 20th century, classical mathematics and Euclidean geometry which is the most intuitive and solid became no longer an adequate basis for architectural design and form seeking process, due to the insufficiency of the traditional classical mathematics with regard to the ever increasing complexity of the world shaped by man. In consequence, the reliance on other branches of mathematics solving this dilemma became a priority.

In particular, the developed calculus during the 18th century had provided mathematicians with tools to develop branches of higher mathematics. These branches incorporated with IT revolution to introduce CAD into architecture design. Then advances in CAD introduce new geometrical potentials breaking away from the canon of the Euclidean geometry and aid the process of form creation. CAD becomes a creative partner with human in the process of form finding.

The reciprocity between developments in mathematics and IT continued. This elaborates new mathematical tools with different activity, rather than creating variants of functional solutions, drafting, modeling and presenting, but also generating forms. This approach is known now by 'Generative Design'. Several generative design paradigms have been proposed, well known tools such as shape grammars, parametric variations and algorithmic generation, and recent tools for searching of form in the world of chaos or random functions like Fractals and Spirolaterals, and evolutionary tools based on the genetic engineering process like Genetic Algorithms and Cellular Automata.

- **<u>Motivation</u>**

Current computer modeling applications are satisfying. However, they need to offer additional options that give designers more form generating capabilities other than the rather tedious current approach.

Though these tools can do everything from generating new forms, ideas and concepts in design, Generative design is still ill defined as a new established research area.

- **<u>Hypothesis</u>**

The study hypothesizes using these tools in generating full three dimensional sculptures or designs that begin to suggest architectural forms. From the preliminary investigations of the resulting forms, there seem to exist great varieties that have that potential.

- **<u>Fields of Study</u>**

- Studying the developments of mathematics in architecture from ancient culture to the twentieth century.

- Studying the enlightenment of generativity in architecture.

- Studying the conventional mathematical generative systems.

- Studying the generative systems in the world of chaos and random functions.

- Studying the evolutionary based generative systems.

- Studying some applications of these generative systems in architecture.

- **<u>Aim (Goal)</u>**

Pointing to and identifying new methodologies or tools that can generate unpredicted novel forms, to trigger inspiration and empower inventiveness during the form seeking dilemma in the architectural design process.

- **<u>Research Objectives</u>**

The main goal is subdivided into the following objectives:

- **<u>Objective 1:</u>**

Describing how Mathematics had been shifted from being a supporting and regulating tool into a creative tool then a generative medium.

- **<u>Objective 2:</u>**

Describing the conventional mathematical generative systems and listing its architectural potentials.

- **<u>Objective 3:</u>**

Explaining how mathematical elements in the world of Chaos and random functions can be interpreted into architectural elements.

- **<u>Objective 4:</u>**

Explaining how evolutionary processes in natural systems can be simulated artificially and interpreted into architectural elements.

- **<u>Objective 5:</u>**

Deducing the characteristics and promises of these generative systems and analytically comparing the efficiency of each generative tool in different applications in the design process.

- **<u>Methodology</u>**

The methodology of the study follows:

A. Historical review for the mathematical utilities in design in ancient cultures until IT evolution in the twentieth century.

B. Analytical study for the conventional mathematical Generative Systems.

C. Analytical study for the advanced chaotic and evolutionary based mathematical generative systems.

D. Deductive study for the characteristics and promises of these generative systems and the difference between the traditional design approach and the generative design approach.

E. Comparative analytical study for the efficiency of each generative tool in different applications in the architectural design process.

- **<u>Thesis Structure</u>**

The thesis consists mainly of five Chapters, as follows:

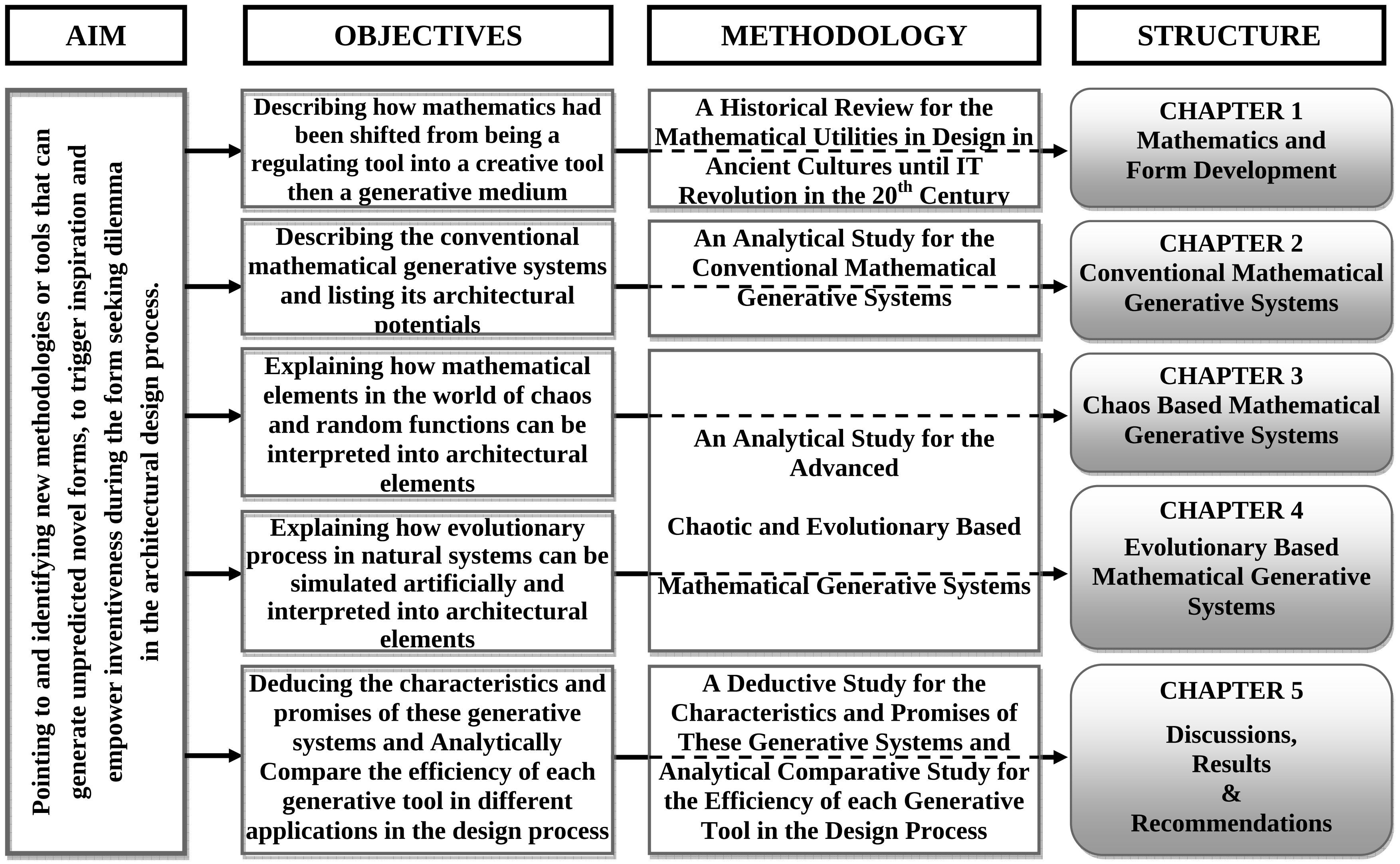
Chapter 1: Mathematics and Form Development.

Chapter 2: Conventional Mathematical Generative Systems.

Chapter 3: Chaos Based Mathematical Generative Systems.

Chapter 4: Evolutionary Based Mathematical Generative Systems.

Chapter 5: Discussions, Results and Recommendations.


AIM
OBJECTIVES
METHODOLOGY
STRUCTURE
Pointing to and identifying new methodologies or tools that can generate unpredicted novel forms, to trigger inspiration and empower inventiveness during the form seeking dilemma in the architectural design process.
Describing how mathematics had been shifted from being a regulating tool into a creative tool then a generative medium
Describing the conventional mathematical generative systems and listing its architectural potentials
Explaining how mathematical elements in the world of chaos and random functions can be interpreted into architectural elements
Explaining how evolutionary process in natural systems can be simulated artificially and interpreted into architectural elements
Deducing the characteristics and promises of these generative systems and Analytically Compare the efficiency of each generative tool in different applications in the design process
A Historical Review for the Mathematical Utilities in Design in Ancient Cultures until IT Revolution in the 20th Century
An Analytical Study for the Conventional Mathematical Generative Systems
An Analytical Study for the Advanced
Chaotic and Evolutionary Based
Mathematical Generative Systems
A Deductive Study for the Characteristics and Promises of These Generative Systems and Analytical Comparative Study for the Efficiency of each Generative Tool in the Design Process
CHAPTER 1
Mathematics and Form Development
CHAPTER 2
Conventional Mathematical Generative Systems
CHAPTER 3
Chaos Based Mathematical Generative Systems
CHAPTER 4
Evolutionary Based Mathematical Generative Systems
CHAPTER 5
Discussions, Results & Recommendations

Chapter 1
Mathematics and Form Development

Chapter 1: Mathematics and Form Development

In most ancient cultures, until the contemporaneous movements, mathematics has been especially relevant in many processes of artistic creation. And many people believe that mathematical thought is an essential element of creativity. Classical mathematics had been the trigger for the geometry of most forms.

In the past, several prominent architects and design thinkers have focused on the issue of the relationship between mathematics and design. The rational forms of Plato and the rules of geometry have formed the basis of antique Greek art, sculpture and architecture and have influenced art and design throughout history in varying degrees. It is common knowledge that ancient Greek and Roman architecture have been based on strong proportioning systems.

These efforts obviously reflect a time when digital technology was not available as a widely used tool and designers did not have the benefit of using such tools. With the increase in computer usage, a greater interest in the relationship of mathematics to art and architecture emerged.

Mathematics and especially geometry have found increasing application in the computer-based design environment of our day. The computer has become the central tool in the modern design environment, replacing the brush, the paints, the pens, the pencils, the compass and the rulers of the artist.

CAD evolution aided the break away from the canon of the orthogonal right angled geometry "Euclidean geometry" and developed ways and tools to handle non-linear complex free forms based on "Non-Euclidean geometry".

Then generative paradigms have been proposed using advanced mathematical principles with the advances IT introduced. And this emerged a new trend of using mathematics in art and architecture which is "Generative art" or "Generative design".

1-1 Mathematics and Form in Ancient Cultures

1-1-1 Nature of Form

The Greek philosopher Plato (427-348 B.C.) asserted that aesthetical forms are based on logical and mathematical rules. Plato had noticed that geometrical forms were "forms of beauty", he had stressed that the use of geometrical forms such as lines, circles, planes, cubes in a composition would aid to form aesthetical forms[1].

Plato described the five regular solids in his Timaeus, and he also illustrated and explained the construction of the five regular solids based on the "most beautiful of all many triangles".

Platonic solids are the three-dimensional bodies whose surfaces consist of identical, regular polygons (e.g., equilateral triangles, squares, and pentagons) which meet in equal angles at the corners. There are five Platonic solids: the Tetrahedron, the Octahedron, the Hexahedron (Cube), the Icosahedron and the Dodecahedron[2] (Fig. 1.1).

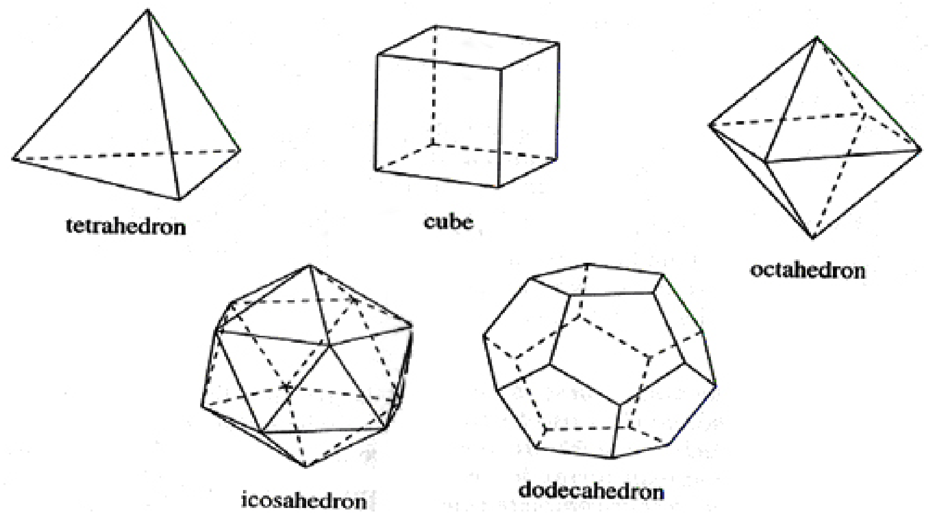


Fig. 1.1: The Platonic solids.

[1] Fowler, D.: *The Mathematics of Plato's' Academy*. Clarendon Press, Oxford, 1999.
[2] Cromwell, P.R.: *Polyhedra*. Cambridge University Press, 1997.

The Greek philosopher found an importance between the Platonic solids and the Empedocles' Four Elements: the fire, the earth, the air, and the water (and the universe).

This association has influenced the philosophers, the artists, and the mathematicians of the Renaissance period. Plato constructed the five solids using simple rules and simple polygons as follows:

"The first will be the simplest" the tetrahedron "which is the original element and seed of fire".

"The second species of solid is formed out of the same triangles". The octahedron "let assign the element which was next in the order of generation to air".

"The third to water" The icosahedron.

"The fourth to earth let us assign the cubical form…to earth is the most immovable of the four" The cube.

"There was yet a fifth combination which God used in the delineation of the Universe" The dodecahedron[1].

The Platonic Correspondences are the following:

Tetrahedron $\rightarrow$ Fire.
Octahedron$\rightarrow$Air.
Icosahedron $\rightarrow$Water.
Cube$\rightarrow$ Earth.
Dodecahedron $\rightarrow$ The Quinta Essentia (the "Universe")

[1] Plato: *Timaeus*. In the Great Books of the Western World. Encyclopedia Britannica, London, Vol. 7, p. 442-477, 1052.

1-1-2 Geometry of Form

- Euclidean Geometry

Euclidean geometry is a geometry based on five postulates, the first four postulates are considered as the postulates of the absolute geometry.

For example, postulate one states that; "a straight line segment can be drawn joining any two points"

In particular the famous, the fifth postulate of "parallelism". States that;" if two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two right angles, then the two lines must inevitably intersect each other on that side if extended far enough".

Euclid's most famous work is The Elements. It contains thirteen books. Book one contains the definitions and concepts used in his work. Here are some of the definitions[1]:

Definition 1: A point is what has no parts.
Definition 2: A line (curve) is length without width.
Definition 3: The ends of a line segment are points.
Definition 4: A straight line is one that lies equally with respect to its points.
Definition 5: A plane has only length and width.
Definition 6: The ends of a plane segment are lines.

Euclid's elements and postulates made a deep impact on the psyche of the western world. Originally viewed as both a tool and a model. For thousands of years, Euclidean geometry, the objects it defined, its postulates, axioms and theorems, were the "truth" in the world of art and architecture, and formed the basis in many forming mathematical elements that appear through the proportions, scale and ordering principles of the form created.

[1] Michele, E.: *Mathland, from Flatland to Hypersurfaces*. Birkhauser-Publishers for Architecture, 2004.

1-1-3 Mathematical Principles and Form[1]

Creating a form in the previous ages depended mainly on many mathematical rules, which the architect used in his designs. These mathematical rules appear through the following:

1-1-3-1 Proportions

Proportioning systems go beyond the functional and technical determinants of architectural form and space to provide an aesthetic rational for their dimensions.

These proportions can usually unify the multiplicity of elements in an architectural design by having all of the design parts belong to the same system of proportions.

Proportions can establish relationships between the exterior and interior elements of a building. The notion of devising a system for design and communicating its means is common to all periods in history. Although the actual system varies from time to time, the principles involved and their value to the designer remain the same.

The most important theories of proportions through ages are:

- **The Golden Section**

Mathematical systems of proportions originate from the Pythagorean concept of "all is number" and the idea that certain numerical relationships manifest the harmonic structure of the universe. The Greeks recognized the dominating role the golden section played in the proportioning of the human body. Believing that both man and his temple should belong to higher universal order, these same proportions were reflected in their temple structures.

[1] Ching, F.D.K.: *Architecture Form, Space and Order*. Van Nostrand Riendold, 1979.

The golden section "can be defined geometrically as a line that is divided such that the lesser portion is to the greater as the greater is to the whole, it can be expressed algebraically by the equation of two ratios: a/b= b/(a+b)"[1] (Fig. 1.2, 1.3).

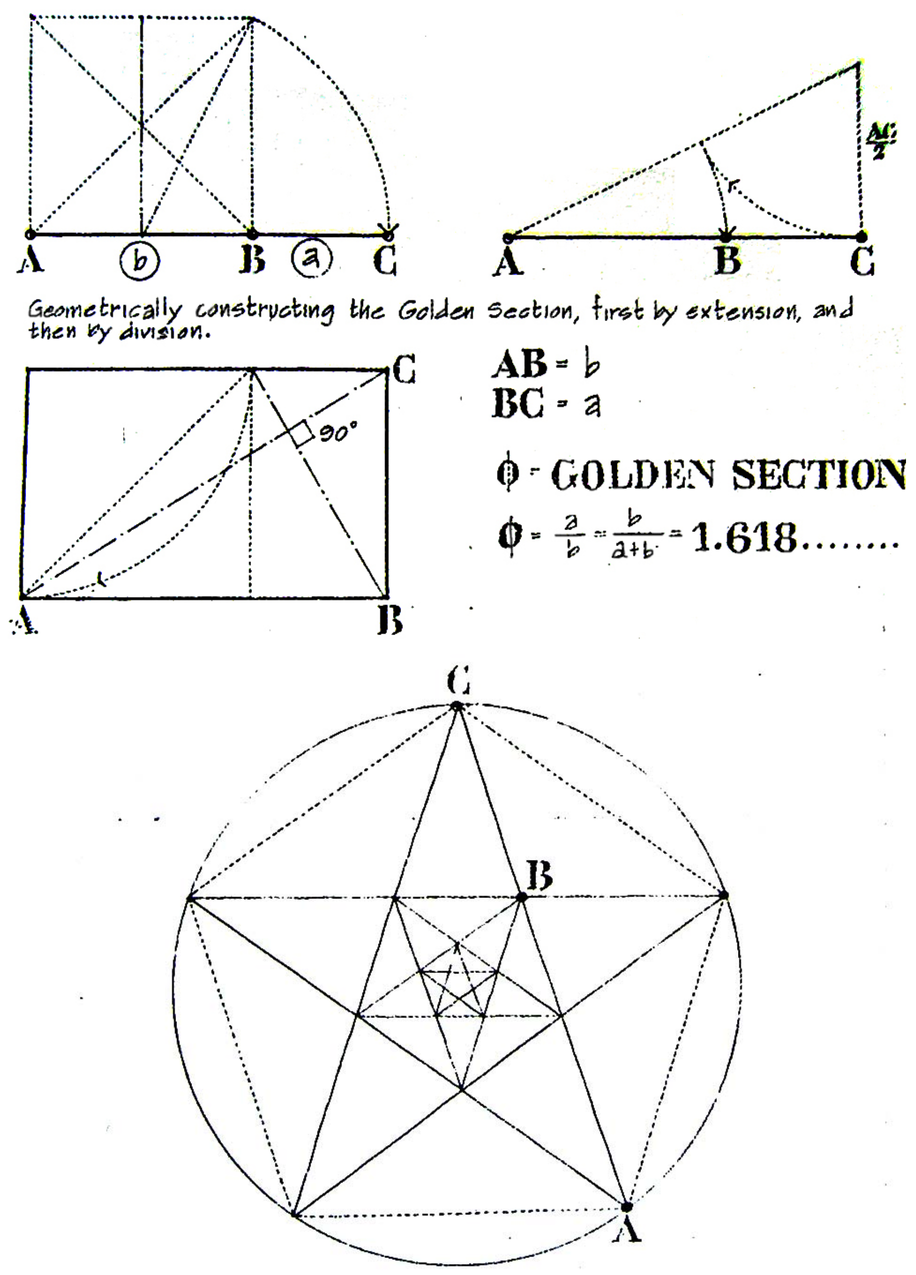

A
b
B
a
C
A
B
C
AC/2
r
Geometrically constructing the Golden Section, first by extension, and then by division.
C
90°
A
B
AB = b
BC = a
ϕ = GOLDEN SECTION
ϕ = a/b = b/(a+b) = 1.618........
C
B
A

Fig. 1.2: The Golden section.

[1] Ibid. p.300.

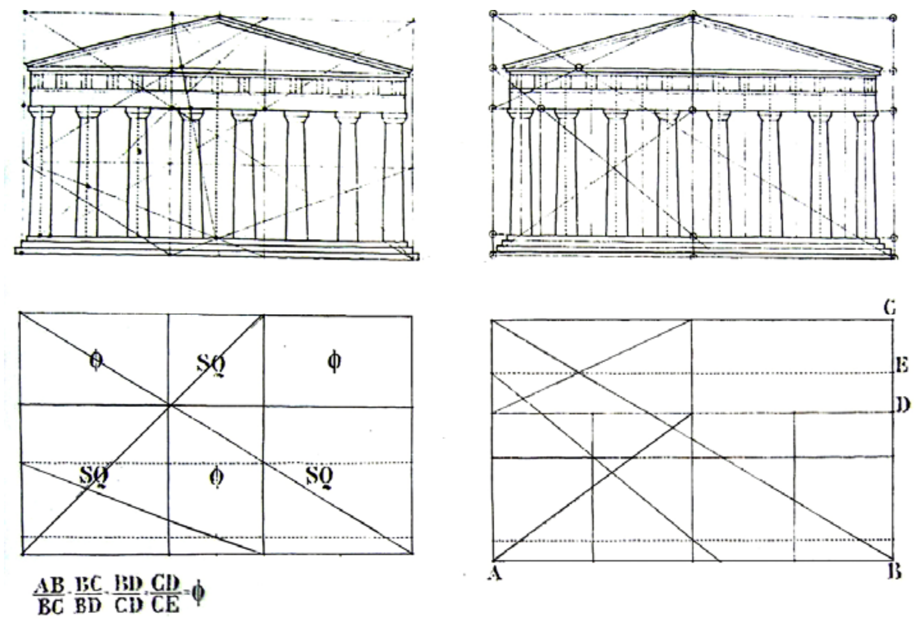

ϕ
SQ
ϕ
SQ
ϕ
SQ
$\frac{AB}{BC} = \frac{BC}{BD} = \frac{BD}{CD} = \frac{CD}{CE} = \phi$
G
E
D
A
B

Fig. 1.3: Two graphic analyses illustrate the use of Golden section in the proportioning of the Parthenon façade (Athens, 447-432 B.C).

It is interesting to note that while both analyses begin fitting the façade into a Golden Rectangle, each analysis then varies from the other in its approach to proving the existence of the Golden section and its effect on the facades' dimensions and distribution of elements.

- **The Orders**

The orders of the Greeks and Romans represented the perfect proportions, which fulfill the expression of beauty and harmony. The basic unit of dimension was the diameter of the column. By using the column diameter as the main unit of module the architects design the capital, the pedestal below, the entablature above, and any other detail in the order. The span between the columns was also based on the diameter of the columns.

Since the size of columns varied according to the size of a building, the orders were not based on a fixed unit of measurement, but were used to ensure that any part of a certain building was proportioned and in harmony the other one.

1-1-3-2 Scale

While proportion refers to the mathematical relationships among the real dimensions of a form or space, scale refers to how people recognize the size of a building element or space relative to the other forms. In visually measuring the size of an element, people tend to use other elements of known-size in their context as measuring devices. These elements are known as scale giving element, and fall into two general categories: building elements whose size and characteristics are familiar to us through experience, and the human figure. In architecture, therefore, there are two types of scale:

Generic scale: The size of building element relative to other forms in its context.

Human scale: The size of a building element or space relative to the dimensions and proportions of the human body.

The designer from a range of choices may predetermine his building elements. Nevertheless, the size of each element is perceived relative to the sizes of other elements around it. For example, the size and proportion of windows in a building façade are usually related to one another, as well as to the spaces between them and the overall dimensions of the façade. If the windows are all of the same size and shape, they establish a scale relative to the size of the façade.

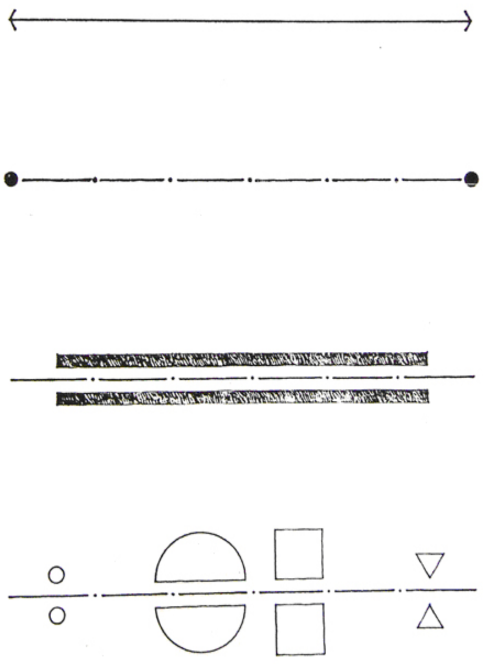
Many building elements have sizes that are familiar to us, and can, therefore, be used to help us gauge the sizes of other elements around them. Such elements as residential window units and doorways can give us an idea of how large a building is, and how many stories it has. Stairs and handrails can help us measure the scale of a space. Because of their familiarity, these elements can also be used to deliberately alter our perception of the size of a building form or space.

1-1-3-3 Ordering Principles

- **Axis**

The axis is the most elementary means of organizing forms and spaces in architecture. Axis is a line established by two points in space and about which forms and spaces can be arranged in a regular or irregular manner. Although an axis is imaginary and not visible, it is a powerful, dominating, and regulating device. Although it implies symmetry, it demands balance. The specific disposition of elements about an axis will determine whether the visual force of an axial organization is subtle or overpowering, loosely structured of formal, picturesque or monotonous[1](Fig. 1.4).

Since, an axis is essentially a linear condition; it has qualities of length and direction, and induces movement and views along its path.



For its definition, an axis must be terminated at both of its ends.

The notion of an axis can be reinforced by defining edges along its length. These edges can be simply lines on the ground plan, or vertical planes that define a linear space

An axis can also be established by a symmetrical arrangement of forms

Fig. 1.4: The axis ordering.

[1] Ibid. p.332.

- **Symmetry**

While axial forms can exist without a symmetrical condition, a symmetrical condition cannot exist without implying the existence of an axis or center about which it is structured. A symmetry condition requires the balanced arrangement of equivalent Patterns of form and space About a common line (axis) or points-center (Fig. 1.5).

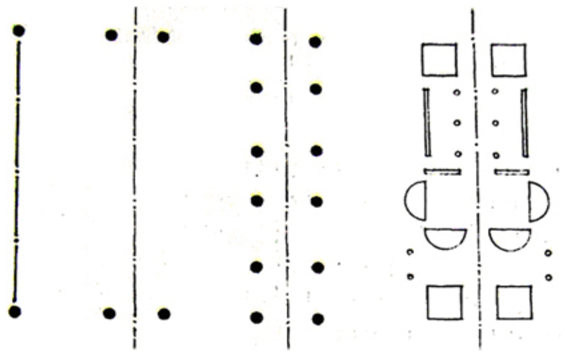


Fig. 1.5: A symmetry cannot without implying the existence of an axis.

There are basically two types of symmetry (Fig. 1.6);
Bilateral symmetry refers to the balanced arrangement of equivalent elements about a common axis.
Radial symmetry consists of equivalent elements balanced about two or more axes that intersect at a central point.

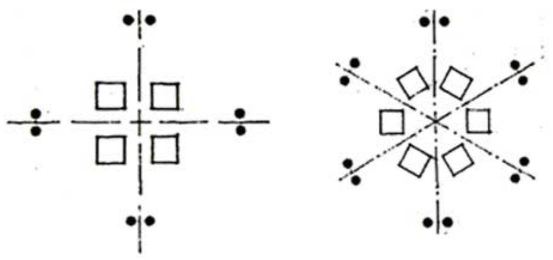


Fig. 1.6: A symmetry could be either a Bilateral or a Radial symmetry.

An architectural composition can utilize symmetry to organize its forms and spaces in two ways. An entire building organization can be made symmetrical. Or a symmetrical condition can occur in only a portion of the building, and organize an irregular pattern of forms and spaces about itself. This latter case allows a building to respond to exceptional conditions of its site or program. The regular, symmetrical condition itself can be reserved for significant or important spaces in the organization[1].

[1] Ibid.

- **Hierarchy**

The principle of hierarchy implies that in most, if not all, architectural compositions, real differences exist among their forms and spaces. These differences reflect, in a sense, the degree of importance of these forms and spaces, and the functional, formal, and symbolic roles they play in their organization. The value system by which their relative importance is measured will, of course, depend on the specific situation, the needs and desires of the users, and the decisions of the designer. The values expressed maybe individual or collective, personal or cultural. In any case, the manner in which these symbolic differences among elements of a building are revealed is critical to the establishment of a visible, hierarchical order among its form and spaces.

For a form or space to be articulated as being important or significant to an organization, it must be visibly unique. This can be achieved by endowing a form or shape with:

- Exceptional size.
- A unique shape.
- A strategic location.

A form or a space may dominate an architectural composition by being significantly different in size than all the other elements of the composition. Normally this dominance is made visible by the variation of size of an element. In some cases, an element can also dominate by being significantly smaller than the other elements of the organization, and placed in a well defined setting[1] (Fig. 1.7).

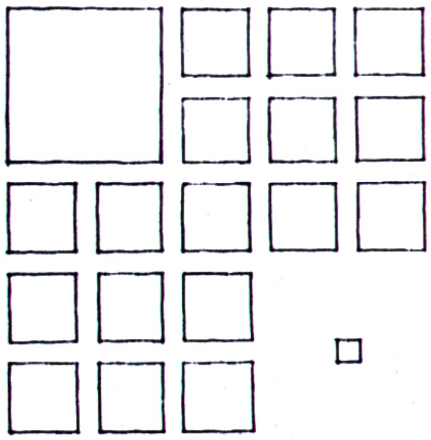


Fig. 1.7: An element dominates by making it smaller than the other objects.

[1] Ibid.

Forms and spaces can be made visually dominant, and thus important, by clearly differentiating their shape from that of the other elements in the composition. A discernible contrast in shape is critical, whether the differentiation is based on a change in geometry or regularity. Of course, it is also important that the shape selected for the hierarchically important element be compatible with its function and Use (Fig.1.8).

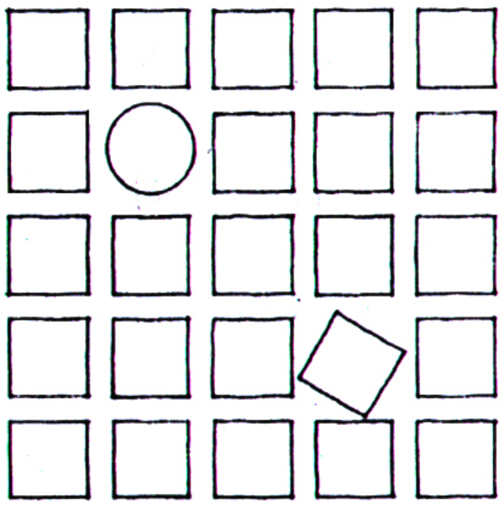


Fig. 1.8: Objects are dominant by shape.

Forms and spaces maybe strategically placed to call attention, themselves as being the important elements in a composition. Hierarchically important locations for a form or space include (Fig. 1.9).

-The termination of a linear sequence or axial organization.
-The centerpiece of symmetrical organization.
-The focus of a centralized Or radial organization offset, above, below or in the foreground a composition[1].

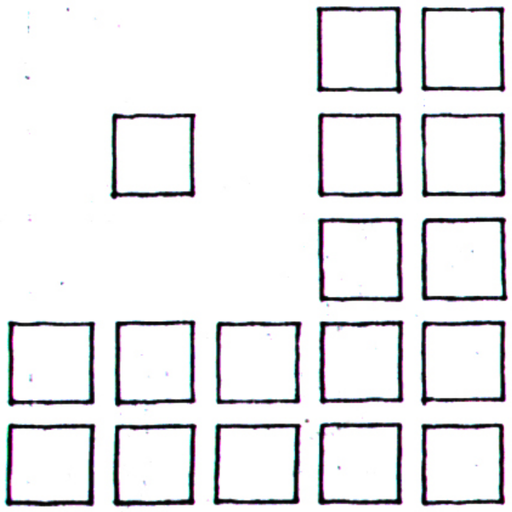


Fig. 1.9: Objects are dominant by displacement.

[1] Ibid. p.339.

- **Rhythm\Repetition**

Rhythm refers to the regular or harmonious repetition of lines, shapes, forms, or colors. It integrates the fundamental notion of repetition as a device to organize forms and spaces in architecture. Beams and columns repeat themselves to form repetitive structural bays and modules of space. Windows and doors repeatedly puncture a buildings' surface to allow light, air, views, and people to enter their interiors. Spaces are repeated to accommodate similar or repetitive functional requirements in the building program. This section discusses the patterns of repetition that can be utilized to organize a series of recurring elements, and the resultant visual rhythms these patterns create. Elements are grouped in a random composition[1] (Fig. 1.10).

1. Their closeness or proximity to one another.
2. The visual characteristics they share in common.

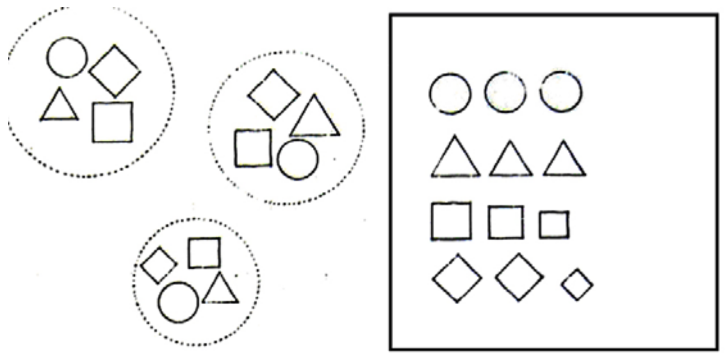


Fig. 1.10: Elements are grouped in a random composition.

The principle of repetition utilizes both of these concepts of perception to order recurring elements in a composition.

The simplest form of repetition is a linear pattern of redundant elements. Elements need not be perfect identical, however, to be grouped in repetitive fashion. They may merely share a common trait, a common denominator, allowing each element to be individually unique, yet belong to same family (Fig. 1.11).

[1] Ibid.

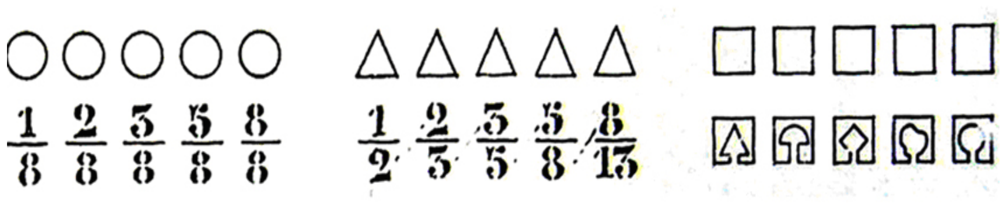


Fig. 1.11: A linear pattern of redundant elements.

Physical traits (Fig. 1.12) by which architectural forms and spaces can be organized in a repetitive fashion are:

-Size
-Shape
-Detail characteristics

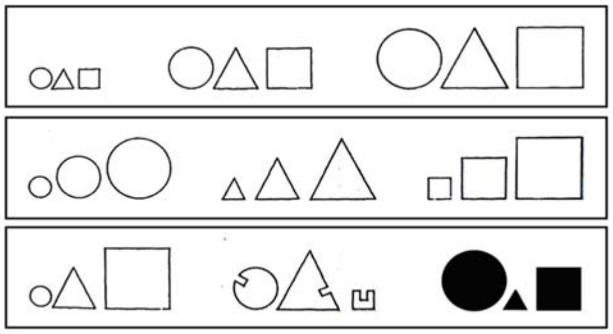


Fig. 1.12: An object is dominant by displacement.

- **Datum**

A datum refers to a line, plane or volume of reference to which other elements in a composition can relate. It organizes a random pattern of elements through its regularity, continuity, and constant presence. For example, the lines of a musical staff serve a datum in providing the visual basis for reading notes and the relative pitons of their tones. The regularity of their spacing and their continuity organizes, clarifies, and accentuated the differences between the series of notes in a musical composition.

In a preceding section, the ability of an axis to organize a series of elements along its length was illustrated. The axis was serving, as a datum. A datum, however, need not to be a straight line. It can also be planar or volumetric in form.

To be an effective ordering device, a datum line must have sufficient visual continuity to cut through or by-pass all of the elements being organized. If planar or volumetric in form, a datum must have sufficient size, closure, and regularity to be seen as a figure that can embrace or gather together the elements being organized within its field.

Given a random organization of dissimilar elements, a datum can organize these elements in the following ways (Fig. 1.13, 1.14, 1.15).

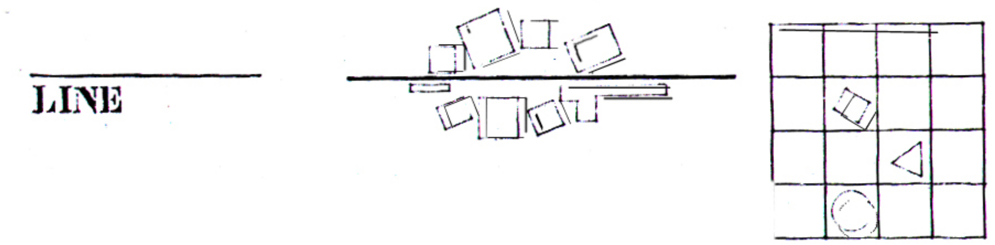


Fig. 1.13: A Line can cut through or form a common edge for the pattern; a grid of lines can form a neutral, unifying field for the pattern.

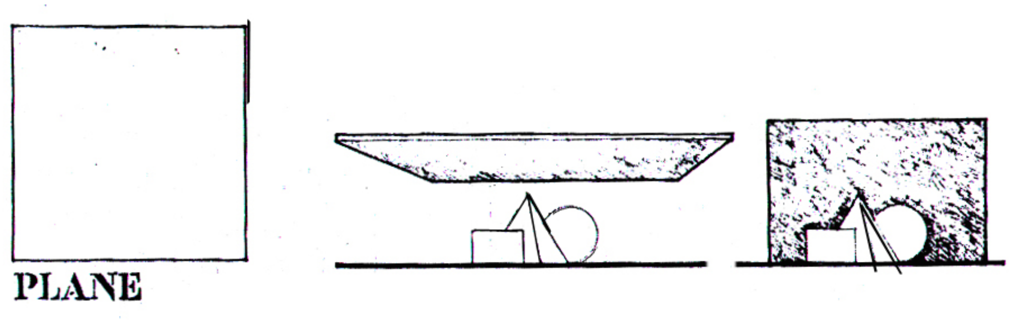


Fig. 1.14: A plane can gather the pattern of elements beneath it. Or serve as a background and in frame the elements in its fields.

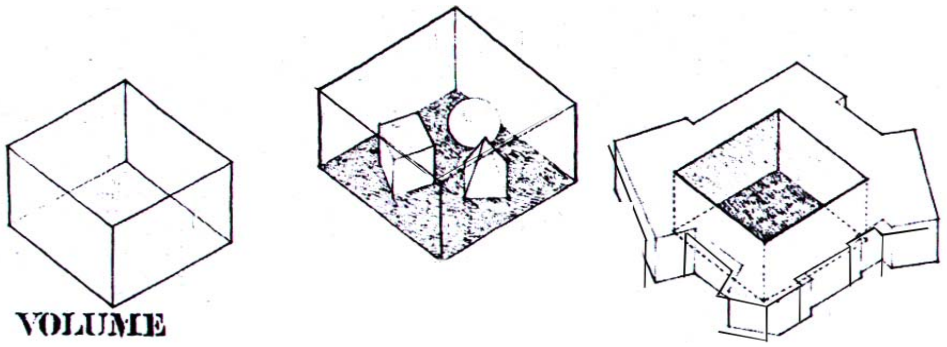


Fig. 1.15: A volume can collect the pattern within its boundaries, or organize them along its perimeter.

- **Transformation**

The study of architecture, as in other disciplines, should legitimately involve the study of its past, of prior experience, of endeavors and accomplishments from which much can be learned and emulated. The principle of transformation accepts this notion[1].

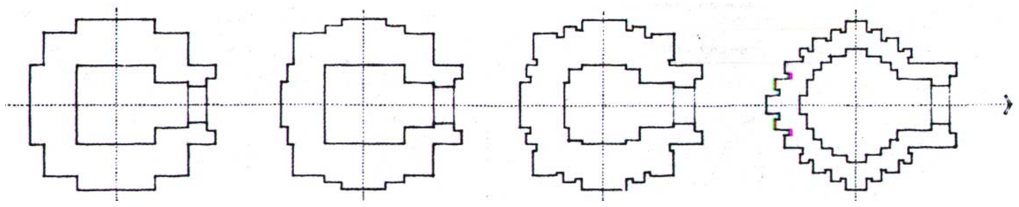


Fig. 1.16: Plan development of North Indian Cella.

The principle of transformation allows a designer to select a prototypical architectural model whose formal structure and ordering of elements might be appropriate and reasonable, and to transform it through a series of discrete manipulations to respond to the specific conditions and context of the design task at hand. Transformation requires first that the ordering system of the prior or prototypical model be perceived and understood so that, through a series of finite changes and permutations, the original design concept can be clarified, strengthened, and built upon, rather than destroyed (Fig. 1.16).

[1] Ibid.

1-2 Mathematics and Form in the 20th Century

1-2-1 Nature of Form

The reliance on classical mathematics and geometry, continued in the beginning of the century due to the inability to perform complex mathematical calculations to create complex forms and the lack of the tool to visualize these complex forms. These factors acted as constraints on the use of higher mathematics and the complex curved forms they described in conceptual architectural design[1].

As a result, conventional box-like, horizontal-vertical, flat-plate architectural forms had been very common in the beginning of the century until modern movements (Fig. 1.17).

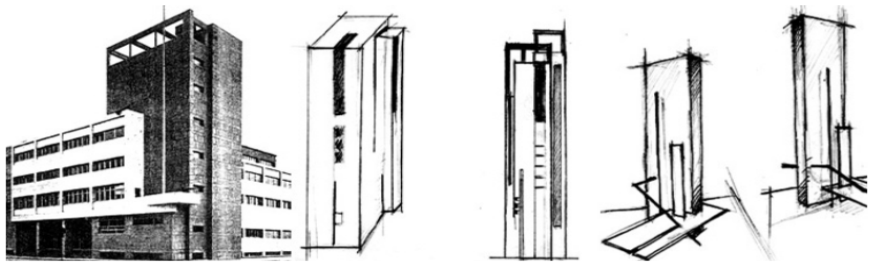


Fig. 1.17: Conventional box-like, horizontal-vertical, flat-plate forms.

On the other hand, philosophy was the key for some freely-formed buildings breaking away from the canon of the traditional classical mathematics without the use of higher mathematics. These achievements have been regarded as more unique and uncommon architectural examples and some of those have even been developed into iconic landmark buildings, either over time such as Sagrada Familia in Barcelona by Gaudi or Ronchamp Chapel by Le Corbusier or the TWA airport terminal in New York by Saarinen. (Fig. 1.18).

[1] Manoa, D.: *Computer Generated Complex Curved Surfaces as an Architectural Design Tool*. A paper presented to the Third International Symposium on Asia Pacific Architecture, 1999.

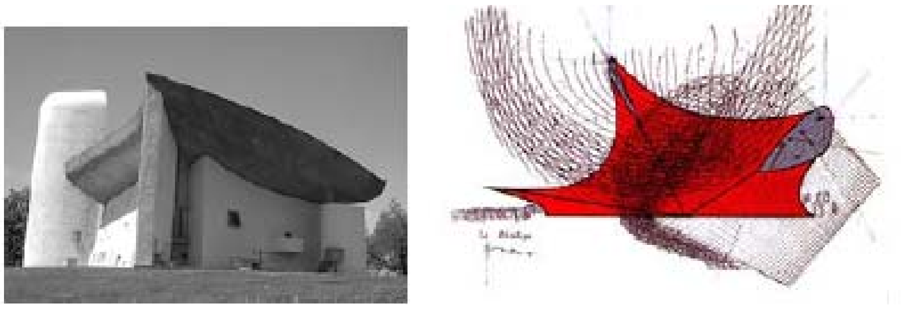


Fig. 1.18: Freely formed building before the era of CAD.

The possibility to create free building forms, have been based on physical models or geometrical construction principles. These forms have in most cases been created with the aid of compasses, curved plastic aids (French curves), pens-ties-to-ropes and clay.

- **Evolution of Computer Aided Design**[1]

In the last decade of the 20th century, classical mathematics and geometry became no longer an adequate basis for architectural design and form searching. Due to the insufficiency of the traditional classical mathematics with regard to the ever increasing complexity of the world shaped by man.

In the beginning of the 1960’s, developments in mathematics combined with developments in computer science, and computer hardware and software, advents the concept of computer aided design (CAD). Computer-aided design (CAD) with all of its aspects has been in an evolution-like changing process since the advent of the concept. This evolution has mainly been based on technology leaps and new innovations in computing and software technology. During the 1980's - the "enlightenment" of computerized design - CAD started to appear more in architectural practice and it also became an acceptable design tool. Finally CAD strengthened its position to be the inevitable architectural tool in design practice during the 1990's.

[1] Pentilla, H.*: Describing the Changes in Architectural Information Technology to Understand Design Complexity and Free-Form Architectural Expression*. Helsinki University of Technology HUT, Department of Architecture, Finland, 2006.

1-2-2 Geometry of Form

The earliest spatial 3D-design with computers in the 1960s had to be simulated with box-like parallel-piped due to limited computing capability. In the 1970s it was possible to model also mathematically defined 3D curved forms with CAD (Fig.1.19), though the geometry of forms didn't break away from the canon of the orthogonal straight lines and traditional curved surfaces[1].

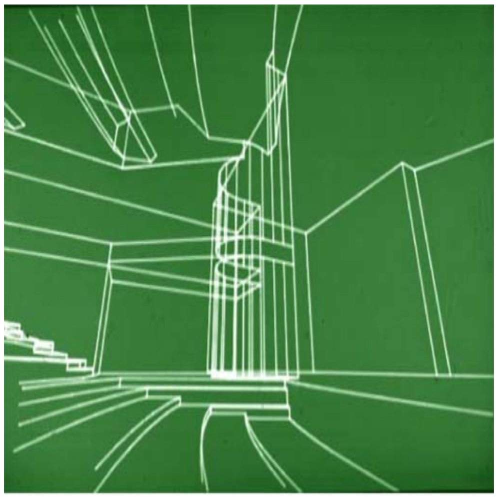


Fig. 1.19: The main hall of Helsinki opera house (architects Hyvämäki - Karhunen –Parkkinen) modelled with Proj-program by Tapio Takala. 1978-84.An early example of free architectural forms expressed with CAD.

Straight and curved lines could be modeled spatially in 3D, though yet without volume modeling.

The description and, consequently, the construction of compound, complex curves were accomplished through concatenating tangent circular arcs and straight-line segments (Fig. 1.20).

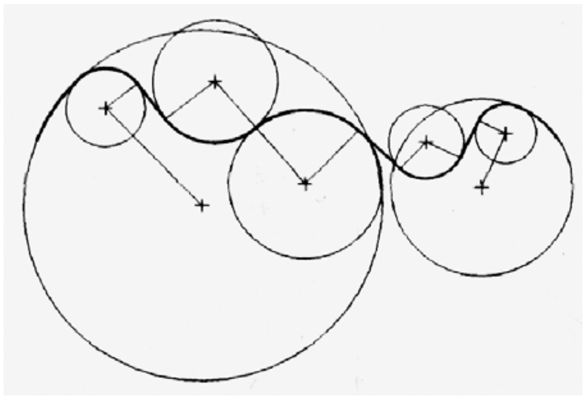


Fig. 1.20: Ordinary method of identifying a curve.q

[1] Ibid. p.398.

Starting from the 1980s and finally during the last decade, The graphic and mathematical functional abilities of the majority of recently used CAD-systems, have developed until now, to handle also non-linear free forms, such as curves and curved surfaces. Current systems balance the needs within architectural design practice, between the Euclidean orthogonal geometry that have been available since the 1960s, and the Non-Euclidean geometry that match the needs of recent architectural expression (Fig. 1.21).

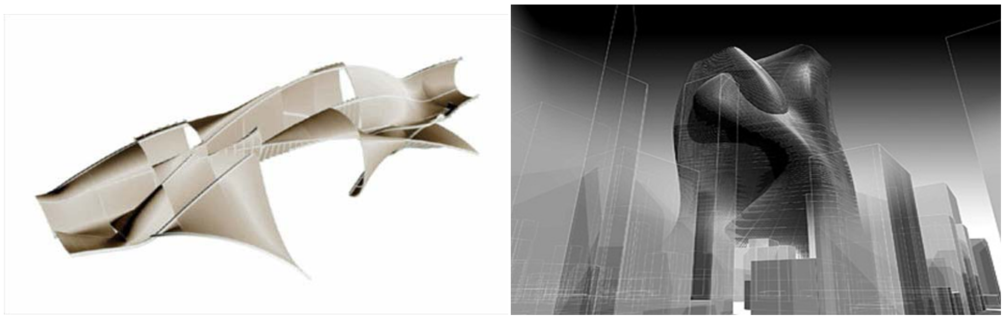


Fig. 1.21: (left). Projects from Greg Lynn Form and Kivi Sotamaa/Ocean-North (Right), Showing the development of forms created in recent CAD systems.

1-2-3 New Geometrical Potentials

Many curve generation techniques have disadvantages when incorporated into an interactive CAD program. Specifically many curve techniques do not give a strong intuitive feel of how to change or control the shape of a curve. Changing the shape of certain curves by moving one or more of the points may produce unexpected, or undesirable, results, both locally and global.

The advent of B-spline curves and then NURBS-surfaces (non-uniform rational B-splines) in CAD-systems has in fact solved this problem and finally released our contemporary forms from the leash of perpendicular axial 3D-shapes[1]. This developed new geometrical potentials for the process of creation of architectural forms.

[1] Chiarella, M.: Geometry *and Architecture: NURBS, Design and Construction*. Proceedings of the Fourth International Conference of Mathematics and Design, special edition of the journal of Mathematics & Design, volume 4, no.1, pp.135-139, 2004.

"Bezier representations of curves and surfaces used in computer graphics were independently discovered by Pierre Bézier an engineer for Renault, and Paul de Casteljau, an engineer for Citroën. Both working for automobile companies in 1970's France, these engineers initially developed a curve representation scheme that is geometrical in construction, and based upon polynomial functions. They extended it to a surface patch methodology that has become the de-facto standard for surface generation in computer graphics"[1].

1-2-3-1 Bezier Curves[2]

It is much easier and appropriate for designers if a curve's shape can be controlled in a predictable way by changing only a few simple parameters. Bezier's curve partially satisfies this need for control.

A Bezier curve in its most common form is a simple cubic equation. Four points define a cubic Bezier curve. Two are endpoints. (x0,y0) is the origin endpoint. (x3,y3) is the destination endpoint. The points (x1, y1) and (x2,y2) are control points. These points, end and control, define what is termed a characteristic polygon (Fig. 1.22).

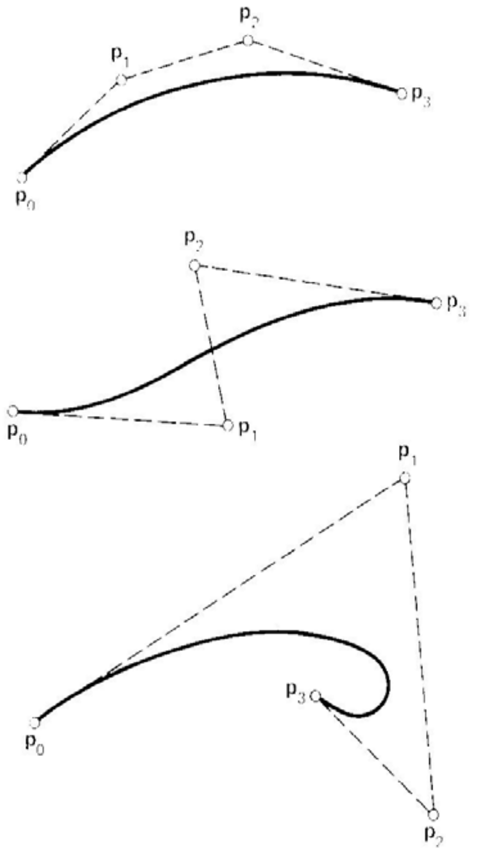


Fig. 1.22: Bezier curves and their Characteristic polyhedrons.

[1] Mortenson, M.E.: *Geometric Modeling*. John Wiley and sons, 1985.
[2] Manoa, D.: *Computer Generated Complex Curved Surfaces as an Architectural Design Tool*. A paper presented to the Third International Symposium on Asia Pacific Architecture, 1999, op.cit.

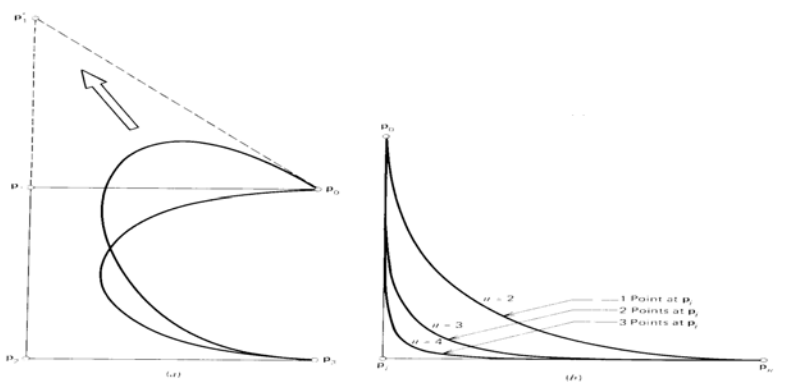


Fig. 1.23: (a) Effects of moving a point on a Bezier Curve and (b) of having coicident points on curves of higher degree.

(Fig. 1.23) shows two examples of cubic Bezier curves, in (Fig. 1.23 a), the placement of the points generates a smooth, uninflected curve, while in (Fig. 1.23 b), the placement of the points generates an inflected curve. There is a facility to modify a curve and the effect can be weaker or stronger depending on the distance or direction the point is moved.

1-2-3-2 Bezier Surfaces (Patch)[1]

Just as the Bezier curve has a characteristic polygon, the Bezier surface has a characteristic polyhedron. Points on a Bezier surface are given by a simple extension of the general equation for points on a Bezier curve (Fig. 1.24).

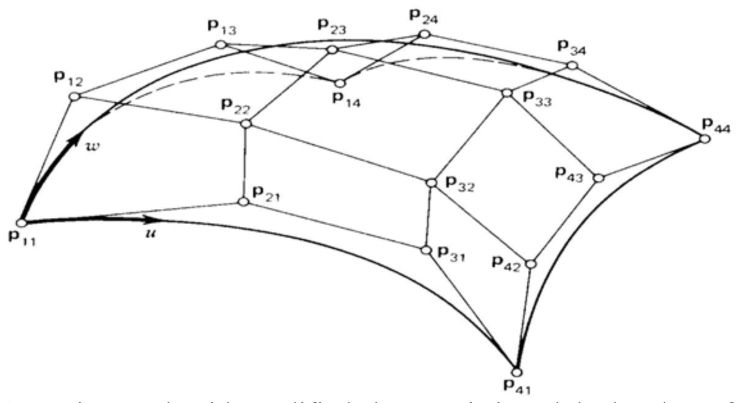


Fig. 1.24: Bezier Patch with modified characteristic polyhedron/net of control points.

[1] Ibid.

The Bezier surface is completely defined by a net of design points describing two families of Bezier curves on the surface. Each curve is defined by a polygon of four points or vertices. (A greater number of points could be used, resulting in a higher degree polynomial).

1-2-3-3 NURBS

The NURBS are mathematical representations of geometry in 3D able to describe any form accurately, from simple lines in 2D, circles, arches, or curves until the most complex solids or organic surfaces in a free way in 3D. Non-Uniform Rational B-splines. NURBS are described as follows:

i. NURBS are a digital equivalent of drafting splines used to draw the complex curves.

ii. NURBS make the heterogeneous, yet coherent, complex forms of the digital architectures computationally possible. NURBS make the construction of these forms attainable by means of computer numerically controlled (CNC) machinery.

iii. The widespread of NURBS is due to its ability to construct a broad range of geometric forms, from straight lines and platonic solids to highly complex sculptured surfaces.

iv. From a computational method point of view, NURBS provide an efficient data representation of geometric forms, using the minimum amount of data for shape computation. For this reason, most of today's digital modeling programs rely on NURBS as a computational method for constructing complex surface models.

The NURBS curves can be changed by manipulating its control points and associated weights and knots, as well as the degree of the curve itself.

The NURBS curves are shaped primarily by changing the location of control points, which do not have to lie on the curve itself, except for the endpoints. Each control point has an associated weight, which determines the extent of its influence over the curve. Increasing the weight of a control point pulls the corresponding curve or surface toward that control point and vice versa (Fig. 1.25, 1.26).

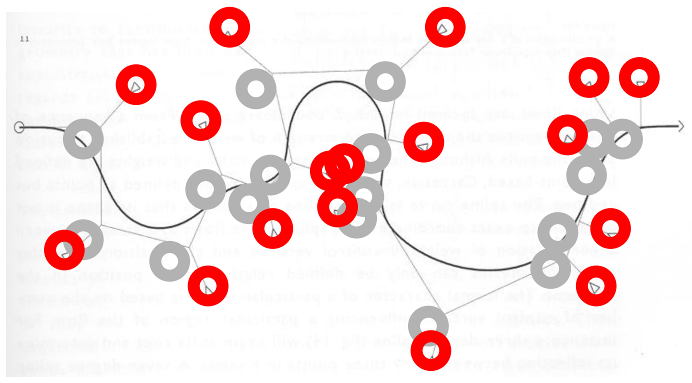


Fig. 1.25: A curve modified by the forces exerted on each control point. (second dark red circles).

When a spline is constructed, the line is not drawn by itself (curved line in image), but rather by the control points (grey circles) that influence the spline.

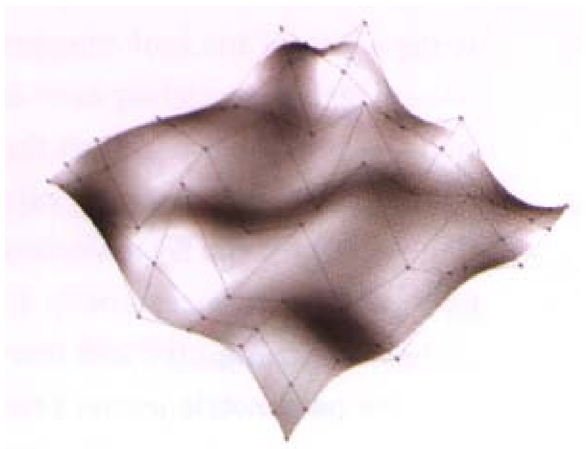


Fig. 1.26: The control lattice for a NURBS surface.

1-3 Mathematics and Generative Design Evolution

The relationship between mathematics, art and design has shifted throughout the last years, Due to developments in mathematics that combined with developments in computer science, hardware and software. These relationships aroused new tools aiding the design process.

Reported attempts to automate the process of layout allocation and space generation started over 40 years ago. Researchers have used several problem representations and solution techniques to describe and generate solutions for the design problem. Several generative paradigms have been proposed using advanced mathematical principles. Generative systems are relevant to contemporary design practice and their integration into the design process allows the development of predicted and unpredicted two and three dimensional design solutions, labor intensive and time consuming to achieve via other methods.

1-3-2 Space Planning Paradigms

1-3-1-1 Graph Theory

The first confirmation existence of generation in design was done by 'Levin' in 1964[1]. Who used an analytical tool box available for the study of complex systems that is rooted in a powerful subfield of mathematics, called "graph theory", which originated in the eighteenth century work of 'Euler'.

The system of elements that interact or regulate each other (a network) can be represented by a mathematical object called a graph. A graph is a collection of nodes and edges: the interacting components of the system are reduced to a set of nodes, and the interactions among the components are represented by edges.

[1] Levin, P. H.: *Use of Graphs to Decide the Optimum Layout of Buildings*. Architect, 14, p.p 809–815, 1964.

Graph can represent any kind of relationship, and architecture is certain kind of relationship, thus, could be explained suitably by graph. Since then, graph theory has been implemented using computer programs in architectural space planning to analyze potentials of individual spaces that compose together a wider system of space[1].

The next figure and table describe basic concept of graph and its geometry. Including 'minimum distance' and 'optimal passage' (Fig. 1.27, Table 1.1).

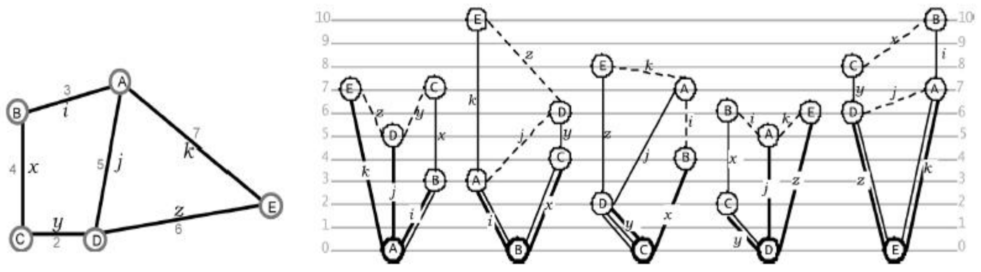
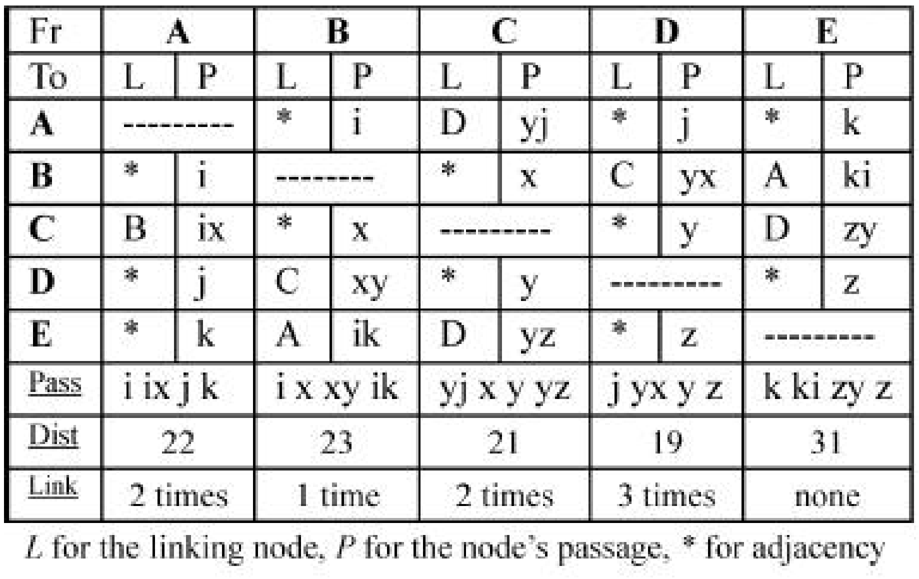


Fig. 1.27: Linear graph diagram.



Fr	A		B		C		D		E	
To	L	P	L	P	L	P	L	P	L	P
A	---------		*	i	D	yj	*	j	*	k
B	*	i	---------		*	x	C	yx	A	ki
C	B	ix	*	x	---------		*	y	D	zy
D	*	j	C	xy	*	y	---------		*	z
E	*	k	A	ik	D	yz	*	z	---------	
Pass	i ix j k		i x xy ik		yj x y yz		j yx y z		k ki zy z	
Dist	22		23		21		19		31	
Link	2 times		1 time		2 times		3 times		none	

L for the linking node, *P* for the node's passage, * for adjacency

Tab. 1.1: Geometric measures of node in minimum-path graph[2].

[1] Kalay, E.Y.: *Modeling Objects and Environment*, John Wiley & sons, 1987.
[2] Nophaket N.: *The Graph Geometry for Architectural Planning*. Journal of Asian Architecture and Building Engineering, May 2004.

More attempts made by 'Grason' in 1970 and 'Steadman' in 1976 to explore the potential for portraying relationships as links within a graph representation. These researchers suggested that the application of this formalism would permit established graph theory algorithms to be implemented within layout generation systems.

(Fig. 1.28) illustrates this representation scheme. In this illustration, a set of adjacency requirements is initially provided by the user (Fig. 1.28 a). Based on these requirements, a graph is constructed where the spaces are represented as nodes and the adjacency requirements are represented as links between the nodes (Fig. 1.28b).

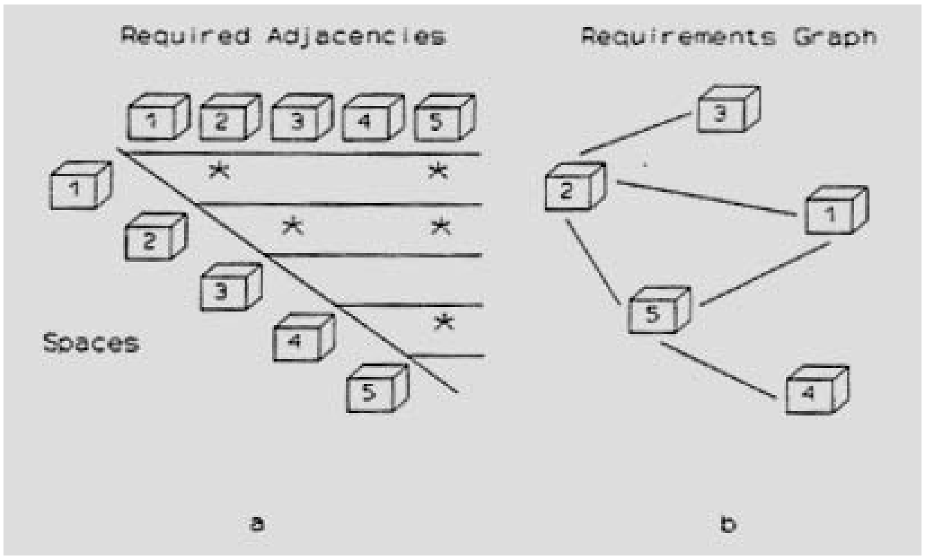


Fig. 1.28: Generating a relationship graph from a relationship matrix.

The graph contains no intersecting links, given this condition. Graph theories prove that all relationships can be accommodated in a 2-dimensional plane. Given the planar requirements graph, a series of potential layouts may be generated which satisfy the spatial requirements (Fig. 1.29a).

Finally, a second graph representation, referred to as a dual graph, characterizes the layout adjacencies and common walls by distinguishing the north-south adjacency links from the east-west adjacency links (Fig. 1.29b)[1].

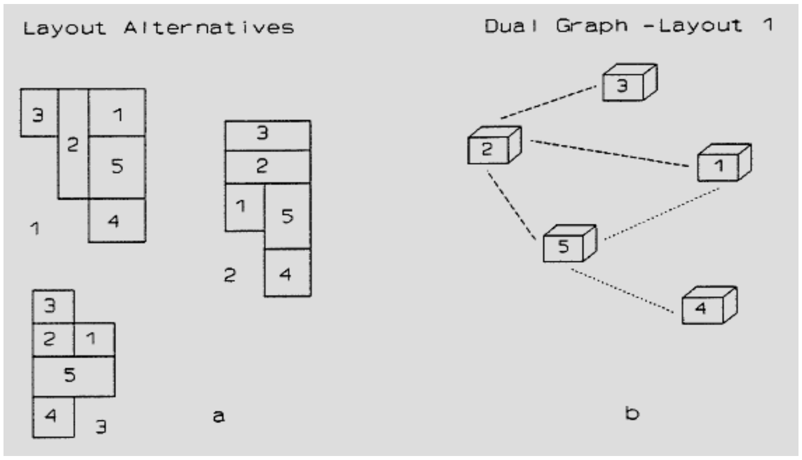


Fig. 1.29: Layout alternatives and a dual graph representation generated from the relationship graph in figure 1.28.

Research efforts founded on this theory have resulted in several layout generation system implementations. Notable among these, are implementations by Grason (Grason 1970), Baybars and Eastman (Baybars and Eastman 1980), hashimshony (Hashimshony and roth 1986), and Rinsma (Rinsma 1988).

Another presented approach was to start with an old solution to a similar space design problem and adapt it to the needs and circumstances of the new problem that differs from the other one. These methods are known as "Case-Based methods".

A third subcategory for automated space planning, capture the knowledge and experience of past designers directly in the form of design rules. These methods are known by "Knowledge-based systems"[2] .

[1] Chinowsky, P. S.: *The CADDIE Project: Applying Knowledge-Based Paradigms to Architectural Layout Generation*. Ph.D. thesis, department of civil engineering, Stanford University, May 1991.
[2] Kalay, E.Y.: *Architecture's New Media. Principles, Theories, and Methods of Computer-Aided Design*. The MIT Press, Cambridge, Massachusetts, 2004.

1-3-1-2 Generative Expert Systems

"Expert systems" are subset of Artificial Intelligence (AI) tools. In these systems, design knowledge is represented within the condition-action formalism of rules. The rules capture the specific conditions under which designers reach decisions for a limited design domain, together with the actions a designer takes when these conditions are present. For example, the following rules capture a design focusing on the placement of two spaces with a required adjacency[1]:

```
IF          a space has been placed in a configuration,
AND         the next space to be placed contains an adjacency
            requirement with the first space,
AND         an available placement position exists
THEN        place the next space in the available position.
```

This use of previous design information includes the adaptation of design concepts, design methodologies, forms, and goals. As designers gain more experience in a given area, successful solutions to previous design problems become prototypes for future problems. Once these prototypes are developed, a designer rarely develops new prototypes due to the extensive knowledge which exists in previous prototypes (Fig. 1.30)[2].

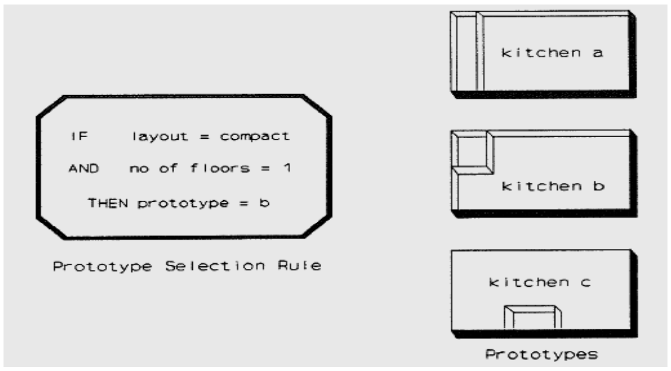


Fig. 1.30: Prototypes refinement rules select appropriate prototypes based on layout conditions.

[1] Chinowsky, P. S.: *The CADDIE Project: Applying Knowledge-Based Paradigms to Architectural Layout Generation.* Ph.D. thesis, department of civil engineering, Stanford University, May 1991. op.cit.
[2] Ibid.

1-3-2 Form Generation Paradigms

Although the previous efforts comprise different methodologies to generate layout alternatives for space planning, no tools were yet developed for the generation of forms in 3-dimensional until the 1970's. Researchers recognized the familial nature of the case-based systems and the kit-of-parts nature of the expert systems as kinds of "languages" and formalized them into systems of rule-based geometrical constructions for designing shapes and forms based on strict compositional rules. This approach came to be known as the "shape grammars" methods.

Advances brought to computational design using higher mathematics and programming languages evolved other tools like: "Parametric variations" and "Algorithmic form generative systems".

Not only conventional mathematics that had been used in developing generative systems, but also chaotic mathematical processes developed new tools like: "Fractals", "Strange attractors[1]" and "Spirolaterals".

The advent of evolutionary design as a result for the reciprocity between evolutionary computations and design, provided researchers with new methods and tools in their quest to rationalize the form generation process. "Genetic Algorithms", "Cellular Automata" and "Artificial Neural Networks" are three tools belonging to evolutionary systems, which are able to generate unexpected novel forms.

The next chapters will summarize and discuss some of these generative systems in detail.

[1] P.S: Some researches classify Fractals and Strange attractors as two generative systems, while other researches grouped them together. In this thesis Strange attractors will be studied as a type of Fractals according to the second opinion.

Summary of chapter one:

Chapter one discussed how the rely on classical geometry -as a branch of mathematics which is the most intuitive and solid in ancient cultures till the beginnings of the 20th century- had been shifted and upgraded into higher branches of mathematics, like the calculus that had been developed in the 18th century. Mathematics development reciprocated with IT and this advents CAD into the architectural design process. CAD developed new geometrical potentials that transformed mathematics from being a supporting and regulating tool that appear only through the proportioning and ordering of the forms produced into a creative tool aided the process of form creation. Researches continued on using other branches of mathematics in architectural design process, particularly in the sixtieth of the 20th century, when graph theory -as another branch of mathematics- had been used in space planning and layout allocation. And this was the enlightenment of the generative design approach in architectural design process.

Chapter 2
Conventional Mathematical Generative Systems

Chapter 2: Conventional Mathematical Generative Systems

Developing the process of form finding demands a shift in the use of mathematics in design and elaboration of new mathematical uses differs from the intuitive use of classical mathematics and geometry.

Mathematical form generation can be used to generate orthogonal conventional forms using basic shape algebra and formal logic or complex three-dimensional curves and folding surfaces using trigonometric parameterized functions or using macro facilities and scripting languages or full fledged programming languages to direct manipulate 2D and 3D forms.

Mathematical form generative systems are well-established themes in computer aided architectural design, including approaches like shape Grammar, parametric variations and algorithmic form generation.

2-1 Shape Grammars

2-1-1 Description

Shape grammars were first introduced by Gips and Stiny in 1972, as an idea to describe visual shape compositions[1]. A shape grammar is a set of rules to specify how one shape or part of a shape can be replaced by another. This simple substitution process can be used to describe a certain design style or generate new ones[2].

[1]Stiny, G.: *Computing with Form and Meaning in Architecture*. Journal of Architectural Education, 39(1): 7-19, 1985.
[2]Knight, T.W.: *Shape Grammars in Education and Practice: History and Prospects.* Online paper, Department of Architecture, MIT, 2000.

In other words, a shape grammar defines a design style in the form of algorithms that manipulate the design components. Several computer systems have demonstrated shape grammars to generate shapes with a certain graphic or architectural style. Much work in shape grammar has analyzed an instance of design style with a set of rules and showed how the rules can be used to generate new designs[1].

- **Types of Shape Grammars[2]:**

i- Standard (Non-Parametric, Basic) Shape Grammars

Standard shape grammar has a set of rules (Fig. 2.1a). In each rule, shape on the left side of the arrow determines which part of the shape will be replaced when the rule is applied. The shape on the right side of the arrow indicates the state after the transformation. An initial shape can be applied by a sequence of rules. (Fig. 2.1b) shows how applying the same simple rule (#2) repeatedly generates a complex pattern. Variation of how many iterations of applying the rule to the initial shape can generate an interesting drawing (Fig. 2.1c).

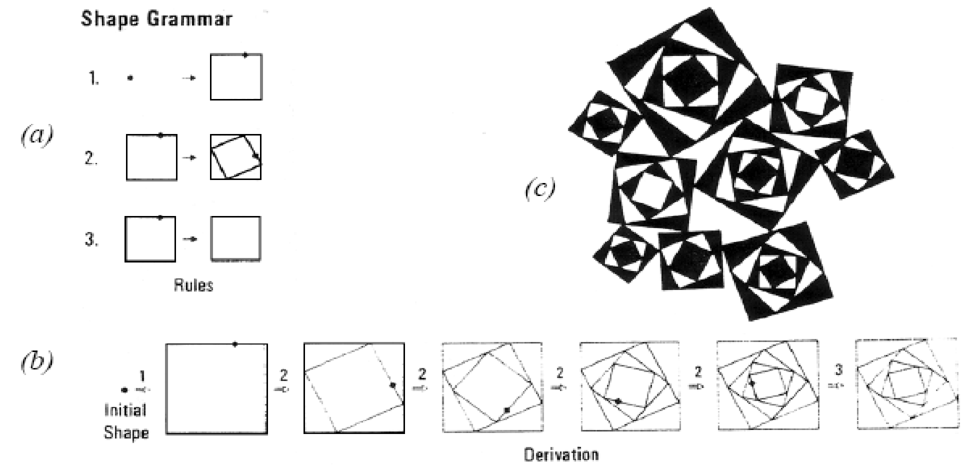


Fig. 2.1: (a) Rules for a Standard Shape Grammar (b) A derivation of the rules (c) A result generated by applying the rules repeatedly. Source: Stiny, 1985.

[1] ibid.
[2]Stiny, G.: *Computing with Form and Meaning in Architecture*. Journal of Architectural Education, 39(1): 7-19, 1985, op.cit.

ii- Parametric Shape Grammars

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 parameters for shape manipulation. Shapes have proportion parameters, and the values of the parameters can be changed. This parametric shape grammar creates shapes with more variation than the standard shape grammar. (Fig. 2.2c) shows the result of a parametric version of the shape grammar shown in (Fig. 2.1b).

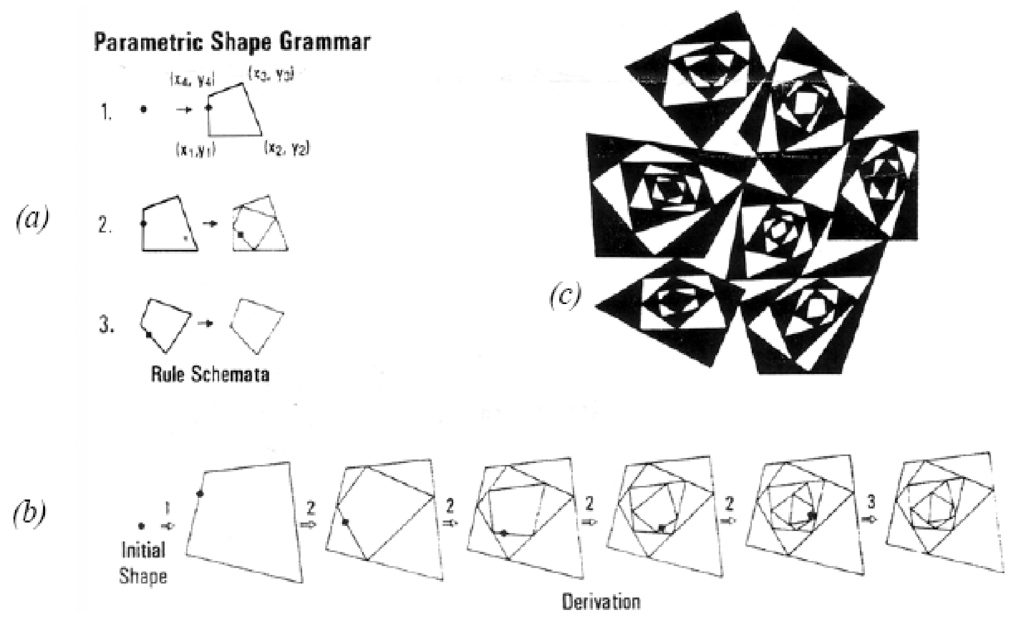


Fig. 2.2: (a) Rules for a Parametric Shape Grammar (b) A derivation of the rules (c) A result shapes generated by applying the rules. Source: Stiny, 1985

Changing a substitution rule of a shape grammar can generate various results. The user arranges shapes using simple operations (translation, rotation, scaling) and defines substitution rules.

Both standard and parametric shape grammars have been used for generating of new forms and analyzing a predefined design styles.

2-1-2 Architecture Potentials

2-1-2-1 Generating Forms

i- 2D Forms

Using shape grammars with a label defined generate a wide variety of 2D forms. The label is a symbol that indicates the orientation of the shape and how to apply this rule in a derivation. The labeled rules show how a rectangle on the left side of the arrow would generate a copy of the shape in a spatial argument.

The two shapes on the right side of the arrow indicate the result of such rules. Left hand side of the rule; an initial shape with a dot label. Right hand side; the arranged shapes illustrating the derivation outcome. With labels at different position, each rule generates a different derivation result (Fig. 2.3, 2.4).

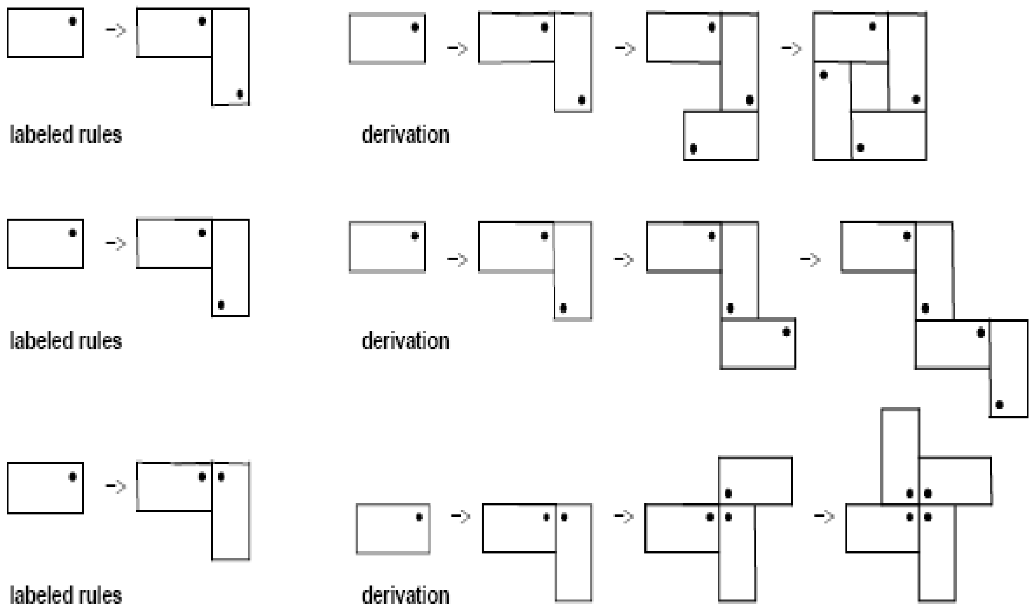


Fig. 2.3: Various 2D labeled rules for standard shape grammar and their derivations. Source: Knight, 2001.

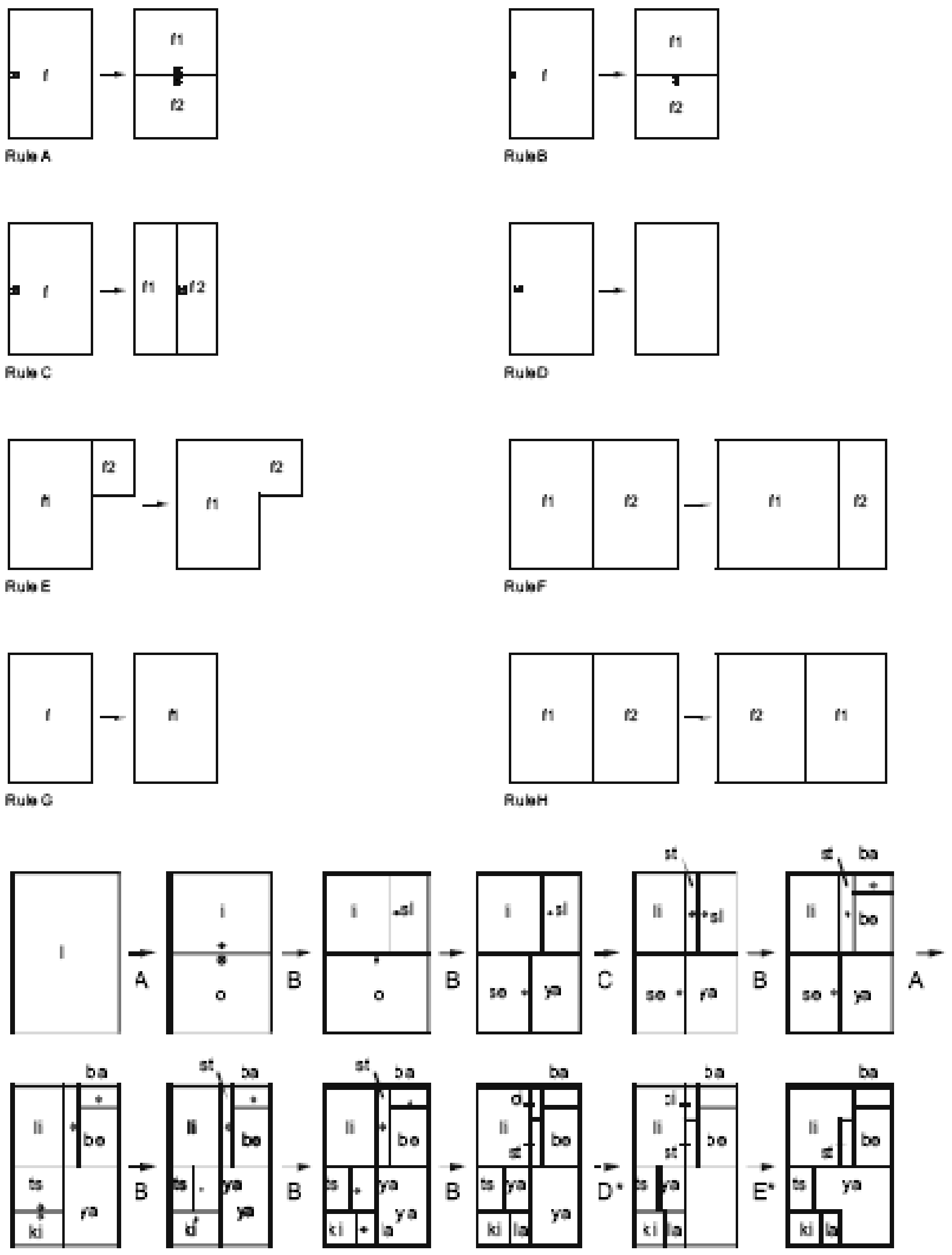


Fig. 2.4: Generating 2 dimensional zoning diagram using labeled rules of shape grammars[1].

---

[1] Heitor, T., Duarte, J.P. and Pinto R.M.: *Combining Grammars and Space Syntax: Formulating, Evaluating and Generating Designs*. The 4th International Space Syntax Symposium, London, 2003.

ii- 3D Forms

3D spatial relations could be configured using the same rules (Fig. 2.5, 2.6).

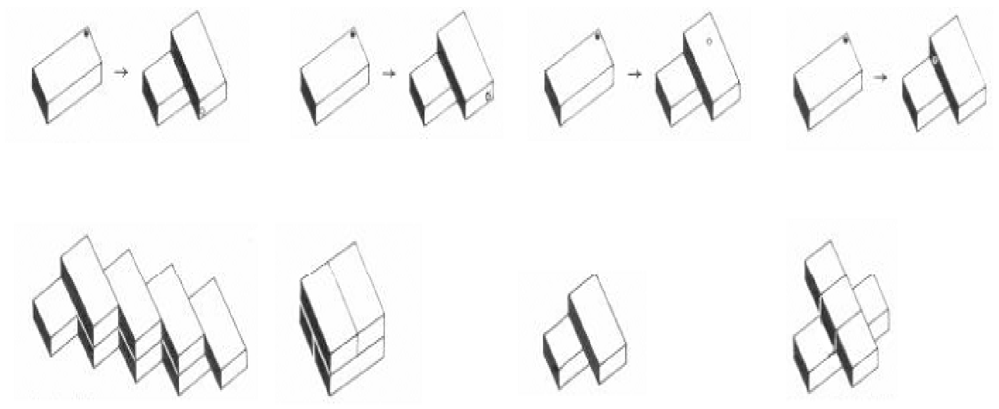


Fig. 2.5: Various 3D labeled rules for standard shape grammar and their derivations.

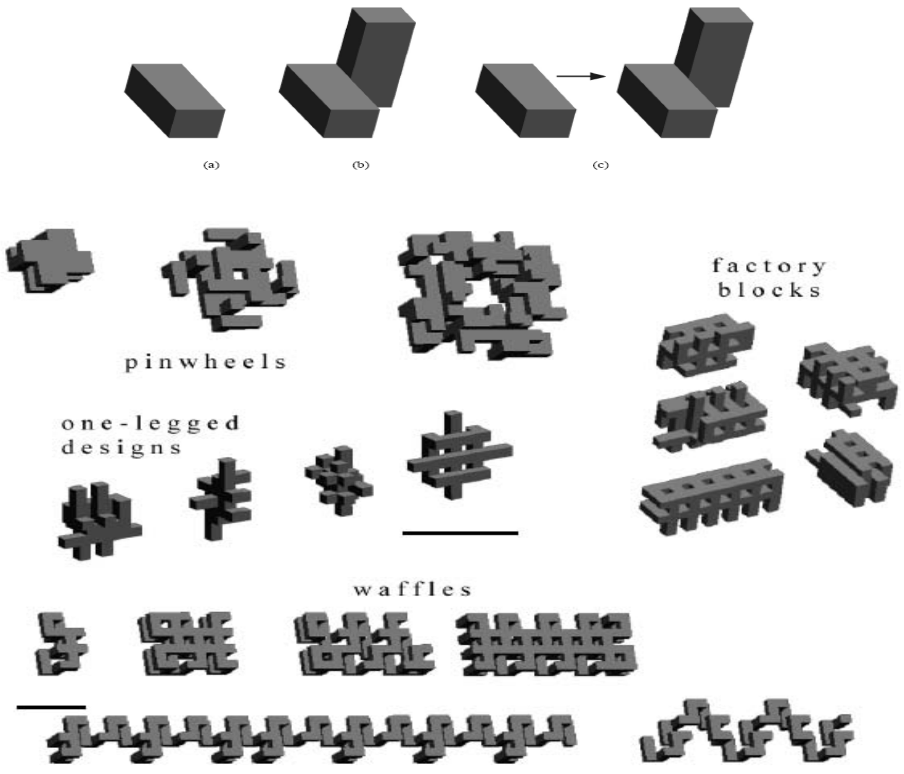


Fig. 2.6: Various 3d forms generated with Shape grammars.

Flemming[1] (1990) presented grammars for various 3D architectural languages: wall architecture, mass architecture, panel architecture, layered architecture, structure/infill architecture, and skin architecture. He used shape grammars to define rules that illustrate how architectural elements can be placed in the space.

(Fig. 2.7a) shows a grammar for wall architecture, one of the languages. The rules of this grammar specify how one wall can attach to another. (Fig. 2.7b) illustrates some wall configurations generated by applying the rules repeatedly. Other architectural languages can be described in a similar way. Designers can use computers to explore and make compositions within these architectural languages.

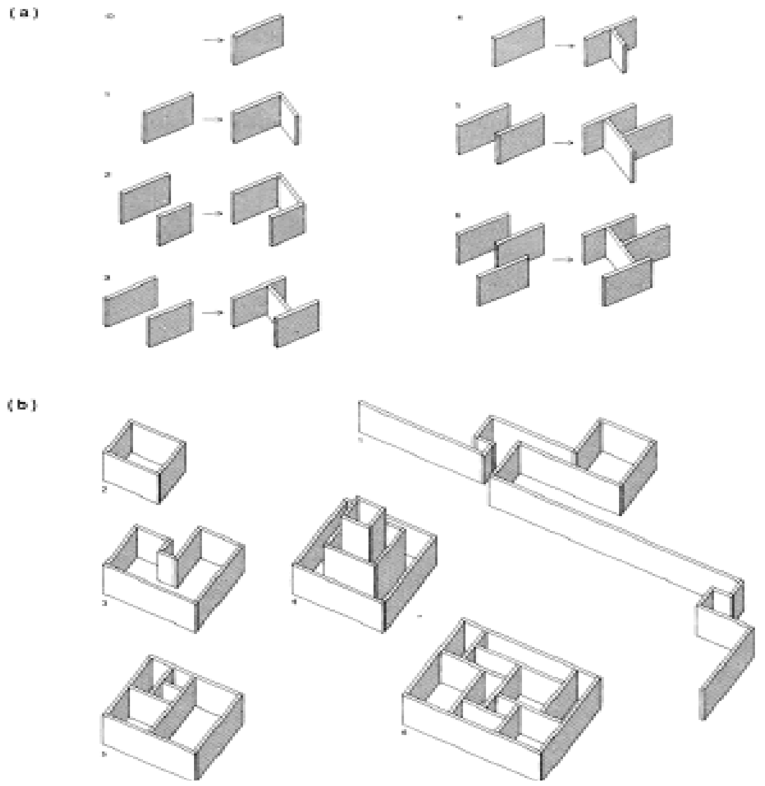


Fig. 2.7: (a) Generation rules for wall architecture (b) Configurations generated by these rules. Source: Flemming, 1990.

[1] Flemming, U.: *Syntactic Structures in Architecture*. M. McCullough, W. J. Mitchell, and P. Purcell, eds., The Electronic Design Studio, The MIT Press, Cambridge, pp. 31-47, 1990.

2-1-2-2 Analysis of Style

There have been several approaches to use a set of rules to analyze and describe a certain design style. For example:

- **Ice-ray Grammar:**

Stiny's Ice-ray grammar is a parametric shape grammar that describes and generates instances of a Chinese lattice design style (Fig. 2.8). This grammar captures the compositional principle of lattice designs into a set of rules[1].

This ice-ray grammar generates various patterns in the Chinese lattice design style (Fig. 2.8a). Looking at Chinese window grilles, Stiny identified four rules for the ice-ray grammar (Fig. 2.8b). Each rule subdivides a shape by inserting a straight line. (Fig. 2.9) shows a derivation to generate a pattern starting from a rectangle shape.

The rectangle is divided into two trapezoids using the third rule, and then the lower trapezoid is divided further into two trapezoids using the third rule. Finally the upper pentagon is split using the fourth rule into a triangle and a pentagon. These subdivisions are applied recursively and generate a pattern in the Chinese lattice design style.

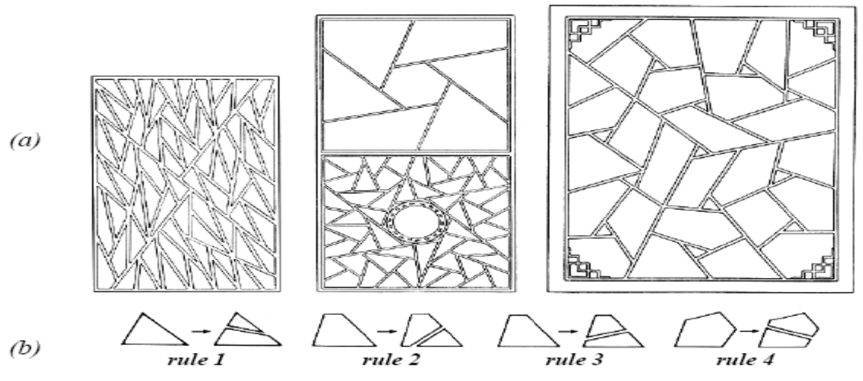


Fig. 2.8: (a) Some results of ice-ray grammar (b) Four rules for a shape grammar of an ice-ray pattern. Source: Stiny, 1977.

[1] Stiny, G.: *Ice-ray: A Note on Chinese Lattice Designs*. Environment and Planning B 4: 89-98, 1977.

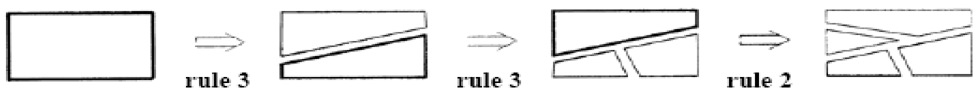


Fig. 2.9: A derivation using the third and fourth rule of ice-ray grammar. Source: Stiny, 1977.

- **Palladian Grammar:**

Stiny and Mitchell[1] (1978) defined a series of rules for villas designed by the sixteenth-century architect, Andrea Palladio. (Fig. 2.10) illustrates some villa plans depicted with the rules. The Palladian villas grammar focuses on describing architectural plans that consist of walls, spaces, windows, and entrances. It has 72 production rules that generate all the villa plans that Palladio designed as well as new ones in the Palladian style. These rules are more complicated than the previous ice-ray grammar.

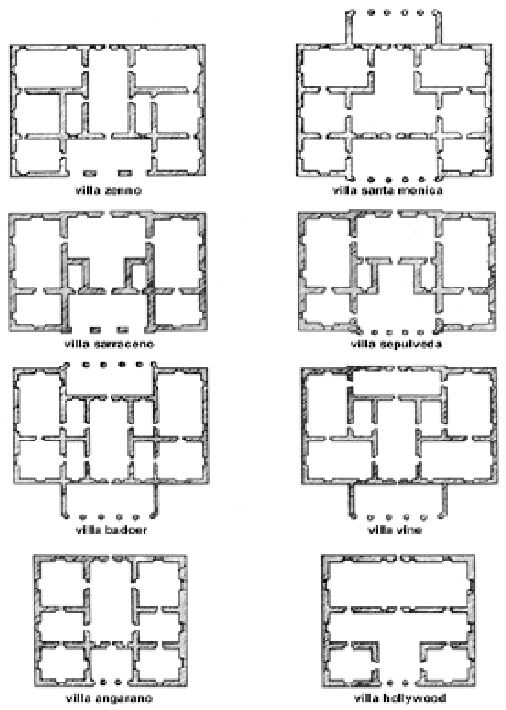


Fig. 2.10: Possible Palladian villa plans with Palladian villas grammar. Source: Stiny and Mitchell, 1978.

[1] Stiny, G. and Mitchell W. J.: *The Palladian Grammar*. Environment and Planning B5, no.1: 5-18, 1978.

(Fig. 2.11) shows how the 72 rules of this grammar are applied to each intermediate drawing and illustrates how Palladio's Villa Malcontenta plan is developed. The grammar starts from defining a single point, which shows a location of the plan on a site. A grid with rectangles is used as an initial layout and controls all subsequent stages of plan generation. The grid is used for generating external walls and rectangular spaces to form rooms in the plan. The principal entrances and columns are then added with windows and doors inserted in the walls to complete the plan[1].

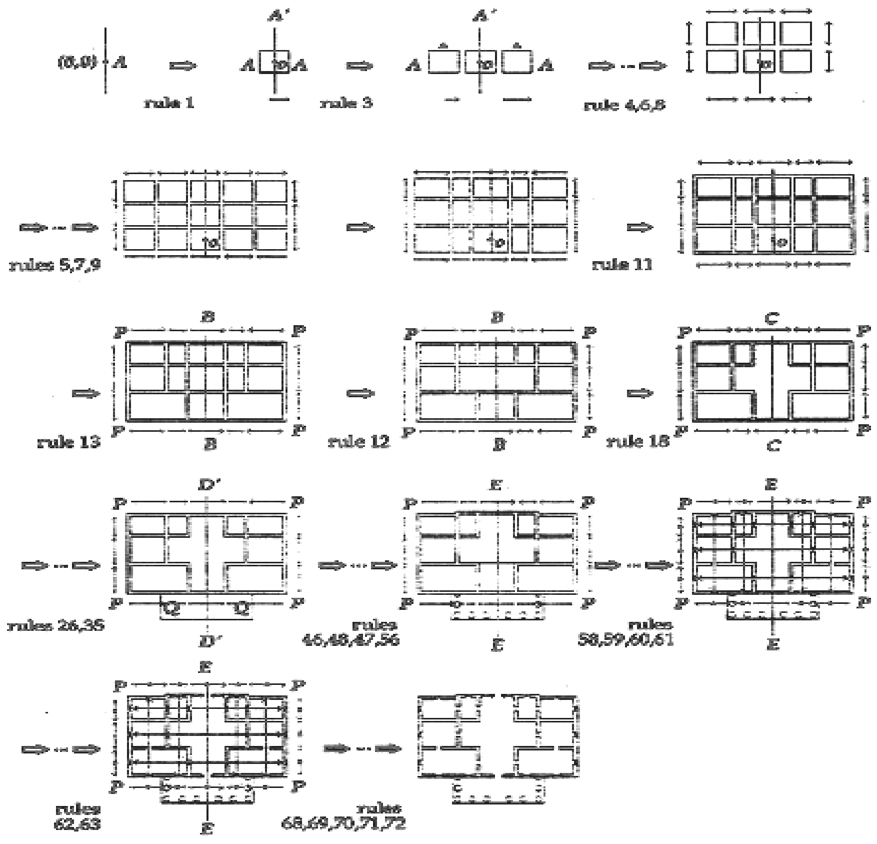


Fig. 2.11: A derivation of Villa Malcontenta using Palladian villas grammar. Source: Mitchell, 1994.

They also implemented the two systems PlanMaker and FacadeMaker to generate various Palladian villa plans and facades.

[1] Mitchell, W. J.: *The Logic of Architecture*. The MIT Press. Cambridge, MA, 1994.

- **Prairie House Grammar:**

Koning and Eizenberg[1] (1981) developed a 3D parametric shape grammar for Frank Lloyd Wright's prairie house style. They used 99 production rules, including 18 rules to arrange major cubic masses and 81 rules to add details to the masses. (Fig. 2.12a) shows one of the massing rules. It extends one mass by attaching another mass to the right side of the existing mass. (Fig. 2.12b) illustrates one of the detailing rules. It adds a terrace object to an existing building.

(Fig. 2.12c) shows the steps of a derivation in 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. 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.

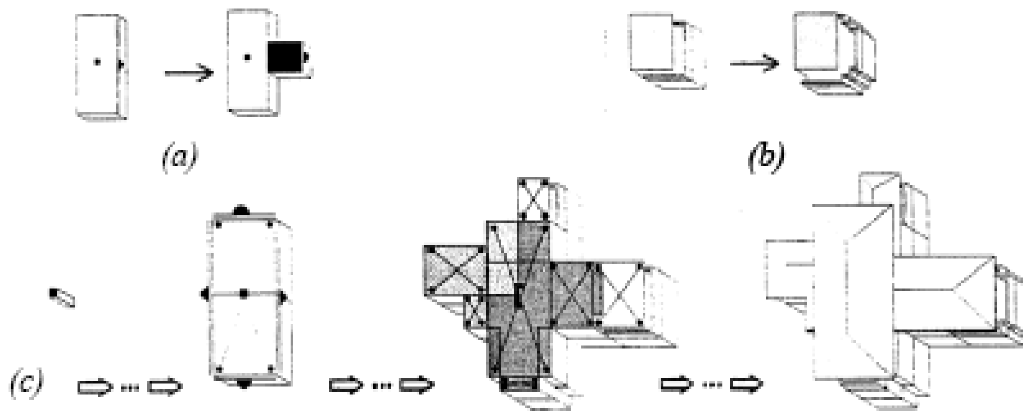


Fig. 2.12: (a) A massing rule (b) Detailing rules for the prairie house grammar (c) A derivation of the rules. Source: Koning and Eisenberg, 1981.

[1] Koning, H. and Eisenburg, J.: *The language of the Prairie: Frank Lloyd Wright's Prairie Houses*. Environment and Planning B8, 295-323, 1981.

There are also rules to add terraces, a basement, and a second story. The final rule completes the generation of the prairie house by adding a roof and chimney. (Fig. 2.13) illustrates some variations generated by the Frank Lloyd Wright's prairie house using the prairie house grammar.

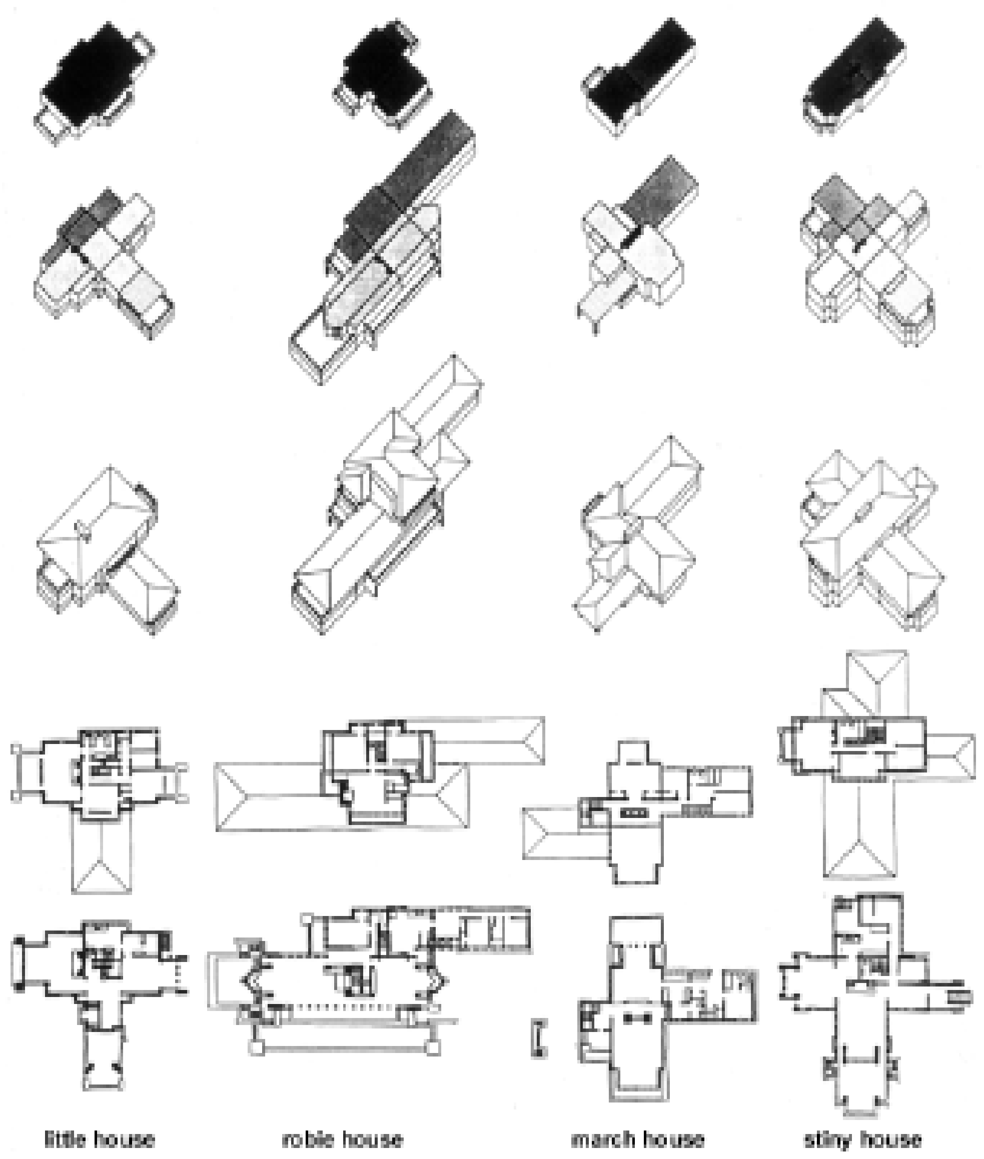
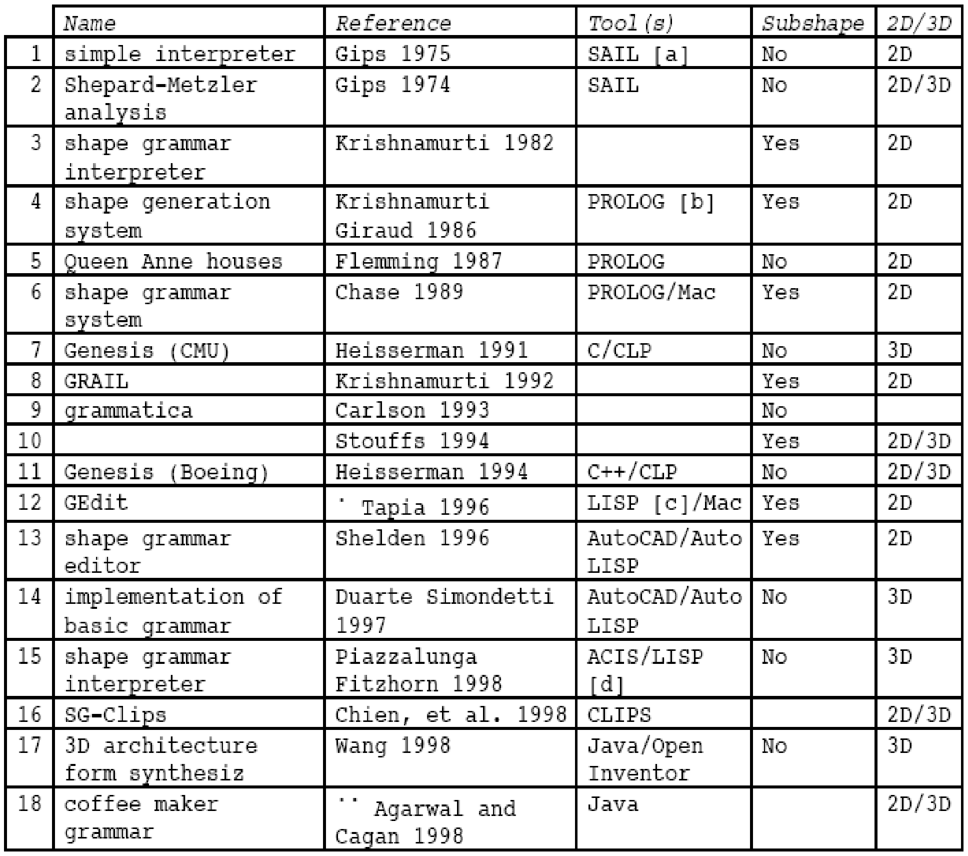


Fig. 2.13: Various results of the prairie house grammar. Source: Koning and Eizenberg, 1981.

2-1-3 Shape Grammar Generative Systems

Many systems of shape grammars had been implemented. (Tab. 2.1) shows a brief survey of shape grammar implementations.



	Name	Reference	Tool(s)	Subshape	2D/3D
1	simple interpreter	Gips 1975	SAIL [a]	No	2D
2	Shepard-Metzler analysis	Gips 1974	SAIL	No	2D/3D
3	shape grammar interpreter	Krishnamurti 1982		Yes	2D
4	shape generation system	Krishnamurti Giraud 1986	PROLOG [b]	Yes	2D
5	Queen Anne houses	Flemming 1987	PROLOG	No	2D
6	shape grammar system	Chase 1989	PROLOG/Mac	Yes	2D
7	Genesis (CMU)	Heisserman 1991	C/CLP	No	3D
8	GRAIL	Krishnamurti 1992		Yes	2D
9	grammatica	Carlson 1993		No	
10		Stouffs 1994		Yes	2D/3D
11	Genesis (Boeing)	Heisserman 1994	C++/CLP	No	2D/3D
12	GEdit	· Tapia 1996	LISP [c]/Mac	Yes	2D
13	shape grammar editor	Shelden 1996	AutoCAD/Auto LISP	Yes	2D
14	implementation of basic grammar	Duarte Simondetti 1997	AutoCAD/Auto LISP	No	3D
15	shape grammar interpreter	Piazzalunga Fitzhorn 1998	ACIS/LISP [d]	No	3D
16	SG-Clips	Chien, et al. 1998	CLIPS		2D/3D
17	3D architecture form synthesiz	Wang 1998	Java/Open Inventor	No	3D
18	coffee maker grammar	·· Agarwal and Cagan 1998	Java		2D/3D

Tab. 2.1: A list of 2d and 3d shape grammar implementations. Available on the web at http://www.shapegrammar.org

2-1-3-1 GEdit

Tapia[1] (1999) developed GEdit, a two-dimensional shape grammar interpreter that provides an interface for users to make or control the rules for spatial layout (Fig. 2.14). A designer arranges shapes and defines the rules in the graphic window. The result of applying the rules is shown in another window.

[1] Tapia, M.A.: *GEdit, A Visual Implementation of a Shape Grammar System.* Environment and Planning B: Planning and Design, vol. 26, pp. 59-73, 1999.

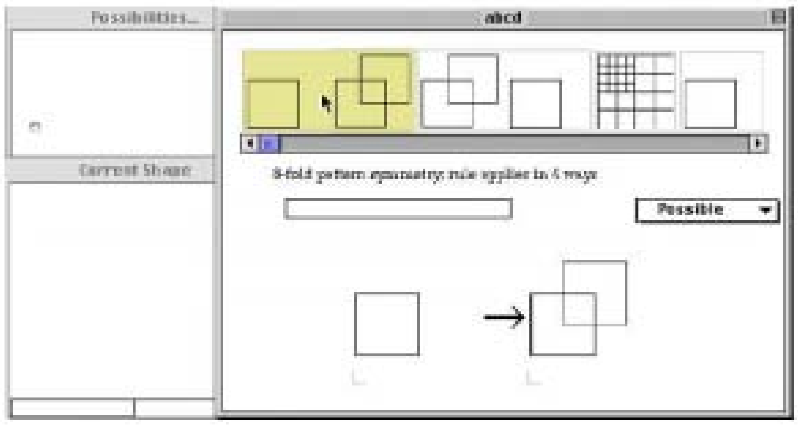


Fig. 2.14: Screenshot of GEdit Interface. Source: Tapia, 1999.

2-1-3-2 Shaper 2D

Mcgill[1] (2001) developed Shaper 2D as an interpreter for standard shape grammar with a graphic interface (Fig. 2.15). It allows designing two rules at the same time with result displayed in the same window. (Fig. 2.16) illustrates the process of applying the generated result to the design process. The designer first generated 2D shape configurations in Shaper 2D (Fig. 2.16a) and then placed it on a site drawing in CAD system (Fig. 2.16b). Finally, the designer further developed the 2D shape into an architectural building plan (Fig. 2.16c).

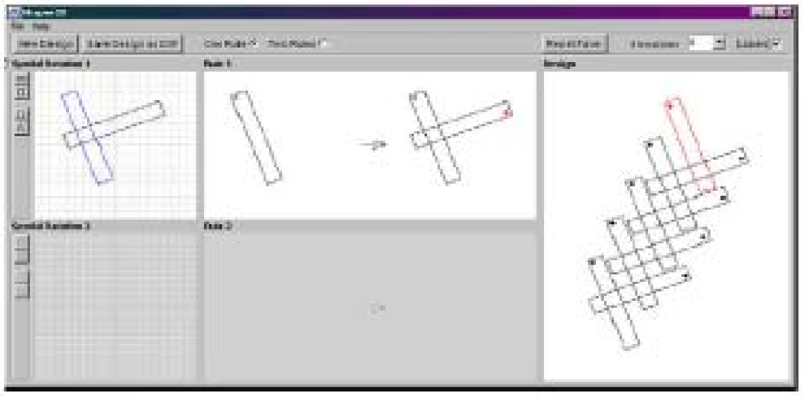


Fig. 2.15: Screenshot of Shaper 2D Interface. Source: Mcgill, 2001.

[1] McGill, M.: *A Visual Approach for Exploring Computational Design*. SMArchS Thesis, Department of Architecture, Cambridge, MA: Massachusetts Institute of Technology, 2001.

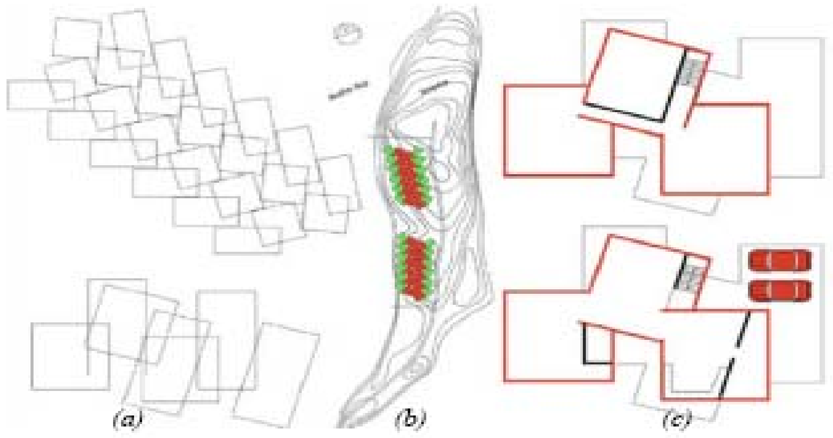


Fig. 2.16: Illustrations for using the result of Shaper2D in the design process (a) The generated result in Shaper 2D (b) Site planning with the result (c) Plan designing with the result. Source: Mcgill, 2001.

2-1-3-3 3D-Shaper

Yufei Wang[1] (1999) developed 3D shaper (Fig. 2.17a) with a dialogue interface for 3D object creation and rule definition. A designer types numerical parameters in the dialogue interface 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 resulting 3D form is then displayed in an Open Inventor Viewer (Fig. 2.17b).

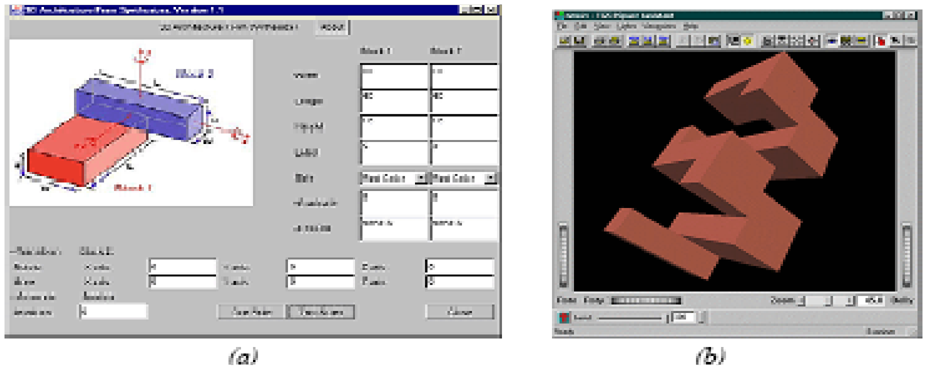


Fig. 2.17: (a) Screenshot of 3D Shaper Interface (b) Screenshot of SGI open Inventor Viewer to see the 3D result of 3D Shaper. Source: Wang, 1999.

[1] Wang, Y.: *3D Shaper, 3D Architecture Form Synthesizer*. SMArchS Paper, Department of Architecture, Cambridge, MA: Massachusetts Institute of Technology, 1999.

2-2 Parametric Variations

2-2-1 Description

Parametric design is turning design into a set of principles encoded as a sequence of parametric equations. The equations are used to express certain quantities as explicit functions of a number of variables[1].

By changing any parameter in the equation new forms and new shapes could be generated. The parameters are not just numbers relating to Cartesian geometry, they could be performance based criteria such as light levels or structural load resistance, or even a set of aesthetic principles.

At the moment, parametric design refers to Cartesian geometry and the ability to modify the geometry by means other than recomposition. The only parameters that can be revised are those that define the measurements of entities and distances along with their relative angles, and the ability to make formal associations between these elements.

Thus the term "parametric design" is more accurately referred to as "associative geometry". Each time a value for any parameter changes, the model simply regenerates to reflect the new geometry.

2-2-2 Architectural Potentials

2-2-2-1 Generating Complex Curves and Ruled Surfaces

A parametric description of form provides high potentiality to generate complex curves and ruled surfaces. Ruled surfaces are able to accomplish high levels of form complexity, especially by their intersections when assembled.

[1]Kolarevic, B.: *Digital Morphogenesis*. In B. Kolarevic, (Ed) *Architecture in the Digital Age, Design and Manufacturing*. New York: Spon Press, 2003.

Ruled surfaces have been extensively applied to architecture, with their potentials adjusted to the new technological means. The Paramorph, designed by DeCoi in 1999, may serve as an example (Fig. 2.18).

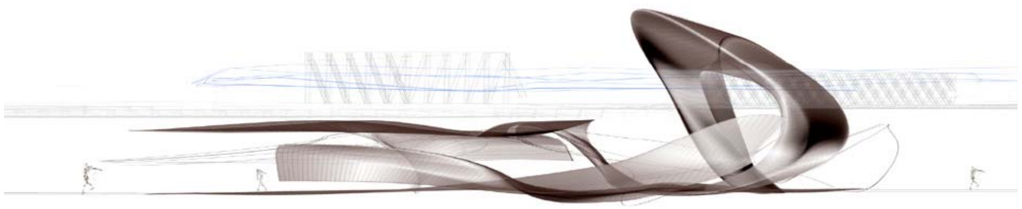


Fig. 2.18: Paramorph (DeCOi 2000).

- **Mathematical Definition of a Ruled Surface**

A 3D surface is called ruled if through any of its points passes at least one line that lies entirely on that surface. A ruled surface results from the motion of a line in space, similarly to the way a curve represents the motion of a point.

A *ruled surface* is a surface swept out by a straight line L moving along a curve b.

Such a surface thus always has a parameterization in *ruled form*

$$x(u,v)=b(u)+v\ d(u) \quad \text{or} \quad x(u,v)=b(v)+u\ d(v)$$

where b is the base curve and d is the director curve. (O'Neill 1966)

Curve b is called Base curve or directrix of the surface. In other words, we get the directrix curve if we fix a point on the moving straight line.

Curve d is called the Director curve of the surface. d(v) is the unit tangent vector with the direction of generator through b. We may visualise d as a vector field on b[1].

[1] ONeill, B.: *Elementary Differential Geometry*. New York: Academic Press, 1966.

(Fig. 2.19) describes the state of some parametric ruled surfaces.

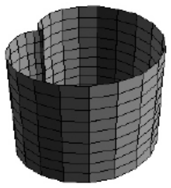
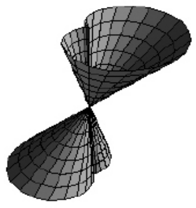
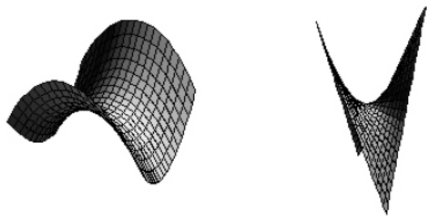
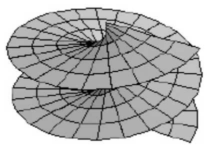
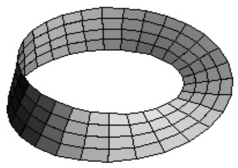
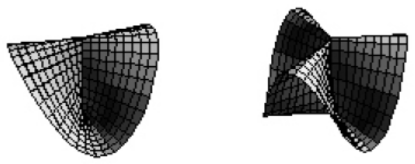
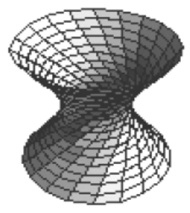
ALL CYLINDRICAL SURFACES  When a straight line moves in the space without changing its direction, the ruled surface it sweeps is called a *cylindrical surface*.	CONICAL SURFACES  When a straight line moves passing constantly through a certain point O the ruled surface it sweeps is called a *conical surface* or a *generalized cone*.
HYPERBOLIC PARABOLOID or SADDLE (a doubly ruled surface)  its vertical cross sections are parabolas, while the horizontal cross sections are hyperbolas.	HELICOID 
	MÖEBIUS STRIP 
PLÜCKER'S CONOID (or CYLINDROID) 	HYPERBOLOID OF 1 SHEET a doubly ruled surface 
Images from *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com	

Fig. 2.19: Ruled surfaces created using parameterized function.
Images from MathWorld-A Wolfram Web Resource. http://mathworld.wolfram.com 9/2007.

The following surfaces (Tab. 2.2) can be expressed parametrically within the following parametric functions:

hyperboloid of 1sheet **cone** **cylinder**	$\begin{bmatrix} a(\cos u \mp v \sin u) \\ b(\sin u \pm v \cos u) \\ \pm c v \end{bmatrix} = \begin{bmatrix} a \cos u \\ b \sin u \\ 0 \end{bmatrix} \pm v \begin{bmatrix} -a \sin u \\ b \cos u \\ c \end{bmatrix}$
hyperbolic paraboloid	$\begin{bmatrix} a(u+v) \\ \pm b v \\ u^2 + 2uv \end{bmatrix} = \begin{bmatrix} a u \\ 0 \\ u^2 \end{bmatrix} + v \begin{bmatrix} a \\ \pm b \\ 2u \end{bmatrix}$
moebius strip	$a \begin{bmatrix} \cos u + v \cos\left(\frac{1}{2}u\right) \cos u \\ \sin u + v \cos\left(\frac{1}{2}u\right) \sin u \\ v \sin\left(\frac{1}{2}u\right) \end{bmatrix} = a \begin{bmatrix} \cos u \\ \sin u \\ 0 \end{bmatrix} + a v \begin{bmatrix} \cos\left(\frac{1}{2}u\right) \cos u \\ \cos\left(\frac{1}{2}u\right) \sin u \\ \sin\left(\frac{1}{2}u\right) \end{bmatrix}$
plücker's conoid	$\begin{bmatrix} r \cos\theta \\ r \sin\theta \\ 2 \cos\theta \sin\theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \cos\theta \sin\theta \end{bmatrix} + r \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$

Tab. 2.2: Surfaces with equivalent parametric equations[1].

2-2-2-2 Simulating Buildings Parametrically

It is possible to create a simple parametric system to generate an almost complete set of building types based on ruled surfaces. Parametric design systems can simulate a wide variety of buildings in a very simple way, in a compact representation than representing a complex building in more detail.

The values of the parameters are presented in 12x n matrices. where n is the number of surfaces. Active parameters of each surface take a real or integer number value, otherwise they are set to 0.

Each surface of a building can be expressed using six parameters, three for the base curve and three for the director curve.

[1] Prousalidou, E.: *A Parametric System of Representation Based on Ruled Surfaces*. Master of Science in Adaptive Architecture&Computation,Bartlett School of Graduate Studi es, University College London, September 2006.

Next, all surfaces had to be assembled in the common framework of the composition. This required a set of transformations (translation and rotation around each axis) to be applied to every surface in order to orient and locate it in the general framework and create the network of interrelated surfaces.

A total number of six parameters, three for translation and three for rotation around each axis- were added to the previous six parameters. It has to be noted that trimming of the surfaces at their intersections is assumed as a precondition for the representation to work out.

When all *n* surfaces are indexed, a 12x n matrix can represent a building[1].

$$\begin{bmatrix} a_{b1} & b_{b1} & c_{b1} & a_{d1} & b_{d1} & c_{d1} & t_{x1} & t_{y1} & t_{z1} & r_{x1} & r_{y1} & r_{z1} \\ a_{b2} & b_{b2} & c_{b2} & a_{d2} & b_{d2} & c_{d2} & t_{x2} & t_{y2} & t_{z2} & r_{x2} & r_{y2} & r_{z2} \\ a_{b3} & b_{b3} & c_{b3} & a_{d3} & b_{d3} & c_{d3} & t_{x3} & t_{y3} & t_{z3} & r_{x3} & r_{y3} & r_{z3} \\ & & & & & \dots & & & & & & \\ & & & & & \dots & & & & & & \\ a_{bn} & b_{bn} & c_{bn} & a_{dn} & b_{dn} & c_{dn} & t_{xn} & t_{yn} & t_{zn} & r_{xn} & r_{yn} & r_{zn} \end{bmatrix}$$

Where:

$$\left.\begin{matrix} \alpha_b \\ b_b \\ c_b \end{matrix}\right\} \quad \text{parameters of [} a\text{*cos(t), } b\text{*sin(t), } c\text{*t] for base curve}$$

$$\left.\begin{matrix} \alpha_d \\ b_d \\ c_d \end{matrix}\right\} \quad \text{parameters of [} a\text{*cos(t), } b\text{*sin(t), } c\text{*t] for director curve}$$

tx : translation of surface around X axis
ty : translation of surface around Y axis
tz : translation of surface around Z axis
rx : rotation of surface around Z axis
ry : rotation of surface around Z axis
rz : rotation of surface around Z axis
n: total number of surfaces that compose the represented building.

[1] Ibid. p.35.

An example showing Los Manantiales building in Mexico simulated parametrically in the figured matrices (Fig. 2.20). Any change in the parameters of the matrix followed by a change in the output or generated form of the building.

- **Restaurant Los Manantiales, Xochimilco, Mexico**

1958 // Architect: Felix Candela

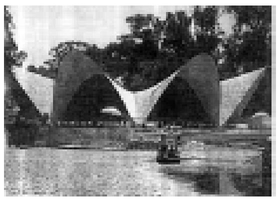


Figure 30. Los Manantiales (R. Bradshaw)

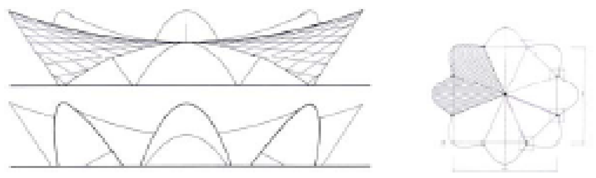


Figure 31. Los Manantiales, elevation and plan (N. Miwa)

$$\begin{bmatrix}
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 0 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 2*PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 3*PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 4*PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 5*PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 6*PI/4 \\
0 & 0 & 50 & -40 & 180 & 0 & 0 & -100 & 10 & 0 & -350 & 0 & PI/8 & 0 & 7*PI/4
\end{bmatrix}$$

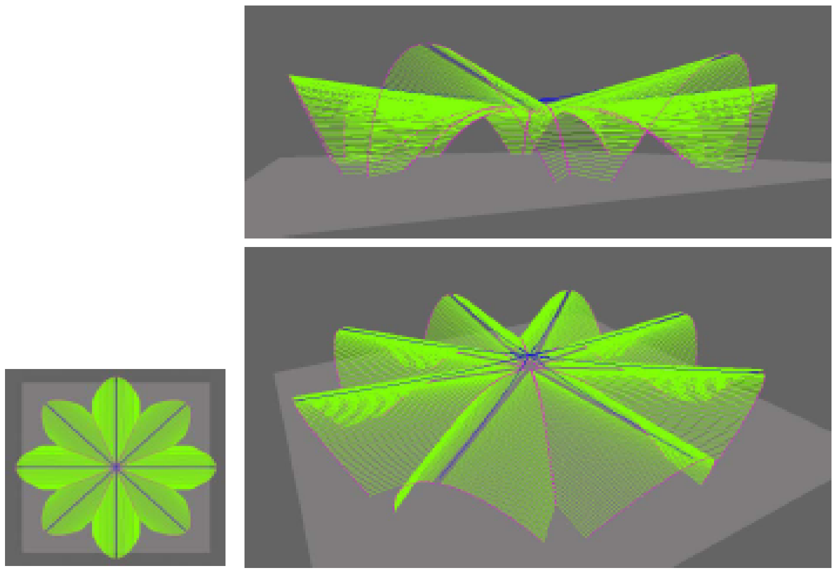


Figure 32: Los Manantiales in Processing.

Fig. 2.20: Los Manantiales represented parametrically in a defined matrix.

2-2-3 Parametric Generative Systems

2-2-3-1 Parametric Ruled Surfaces[1]

The concept of the system is based on the decomposition of the structure into elementary units, i.e. a number of surfaces. Once the surfaces were generated, then assembled to form the architectural composition. The ruled surfaces had to be modeled as a set of parameters so that values could be assigned to the parameters in accordance to the given requirements. Specifying the parameters was the starting point of the system investigation. The use of parametric equations was obviously favored against algebraic expression of the surfaces. Mathematical equations were expressed as position vectors.

The construction of a surface required two parameterized curves, i.e. the base curve and the director curve.

The Base curve: (directrix) is the curve along which runs the straight line (ruling or generatix). The parameter *t* gives consecutive points on this curve. The points form one edge of the line which extends in the direction given by the director curve.

The Director curve: may be understood as a given sequence of unit vectors that varies continuously with *t*. the surface is swept out by moving a straight line along the parameterized curve c so that it points in the direction *z(t)* at each time *t*.

Looking at the parametric functions of those surfaces, it was observed that the most of them are expressed in terms of the sine and the cosine functions and appear to have an underlying relationship. The relation ship was further examined and a significant point was revealed.

The parametric function of the helix expressed in vector form is:

[a*cos(t), b*sin(t), c*t]

[1]Ibid.

If the z vector component is suppressed the curve is degenerated to a **circle**.

If the x vector component is suppressed the curve is degenerated to the parametric **sine** function

If the x and y vector components are suppresses the curve is degenerated to a **line** (Tab. 2.3).

Curves represented as Vectors of trigonometric functions	
Line	[0, 0, c*t]
Circle	[a*cos(t), b*sin(t), 0]
Helix	[a*cos(t), b*sin(t), c*t]
Sine	[0, b*sin(t), c*t]

Tab. 2.3: Curves represented using trigonometric functions[1].

The combination of these four curves as base and director respectively can generate sixteen ruled surfaces. The rotation of one curve around the axes generates thirty two extra surfaces (sixteen for each rotation).

Among these forty eight surfaces the plane, the cone, the cylinder, the hyperboloid and the helicoids are included, in addition to the commonly used sinusoidal surface. The wide spectrum of generated surfaces supported the decision to build the system of representation based on these 4 curves or the helix and its degenerated expressions.

The various forms generated in this way are presented in the following Figures (Fig. 2.21, 2.22, 2.23). In all figures, the base curve is constantly oriented along the z-axis while the director curve is rotated each time.

[1] Ibid.

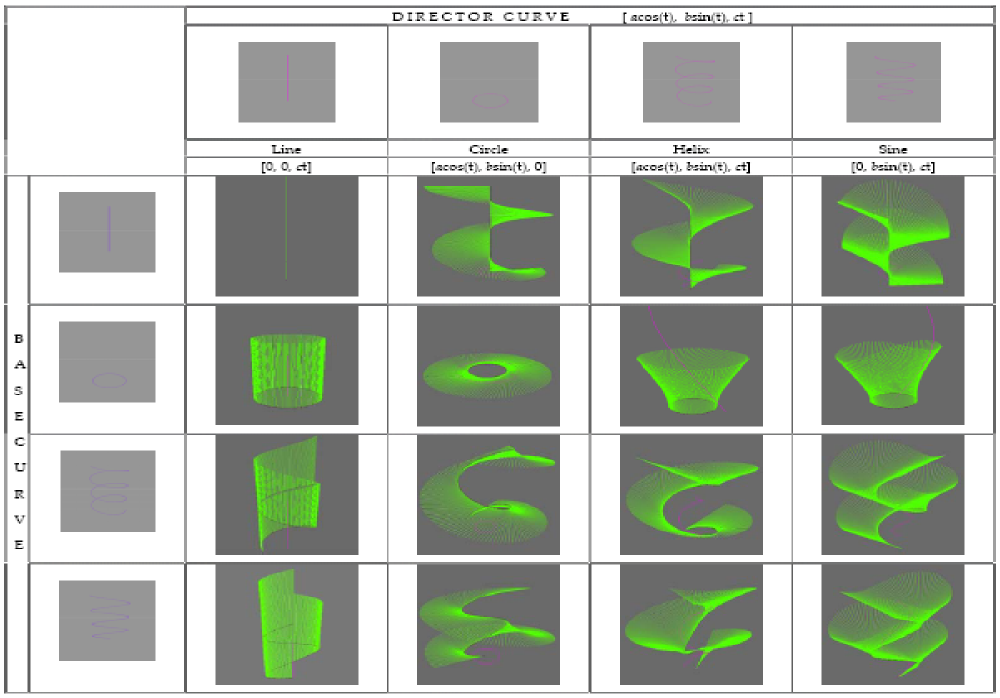


Fig. 2.21: Surfaces generated using parameterized function system.

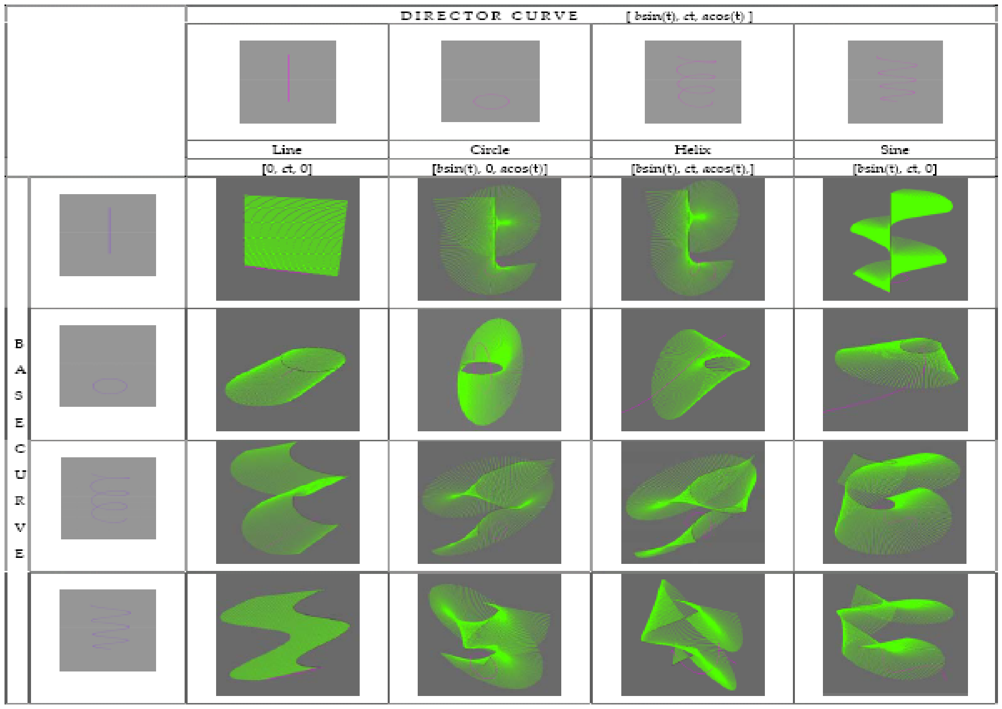


Fig. 2.22: Surfaces generated using parameterized function system.

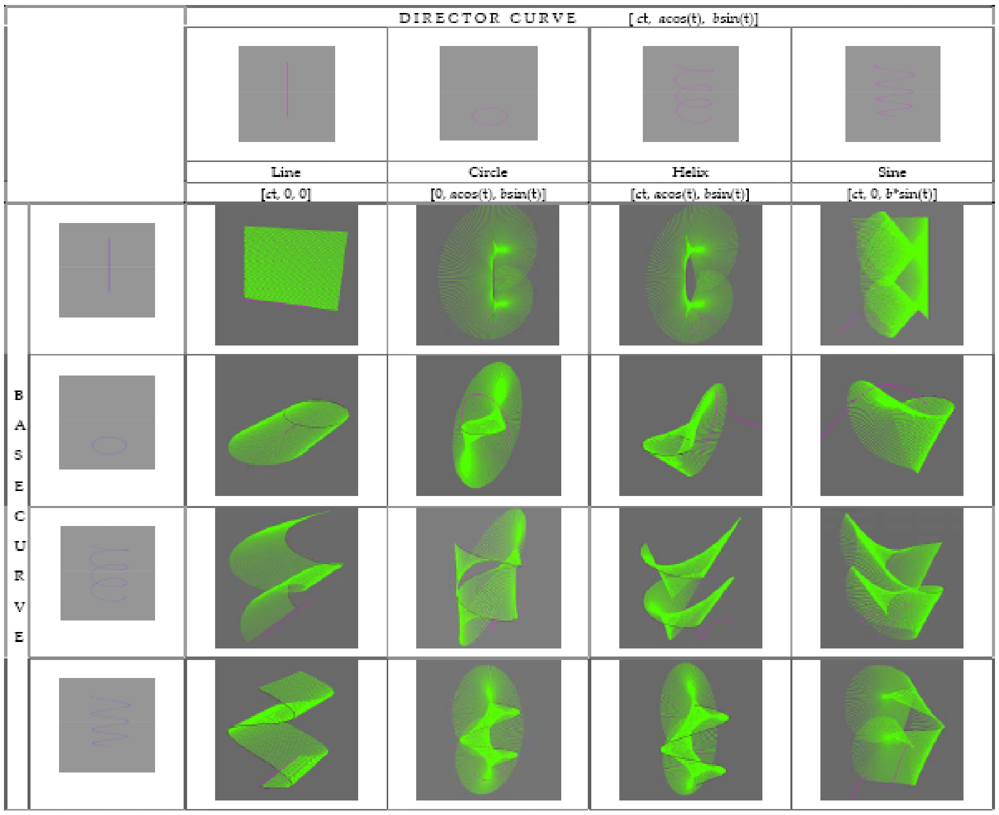


Fig. 2.23: Surfaces generated using parameterized function system.

(Fig. 2.24) shows composed surfaces with equivalent parametric functions. The program employs some additional variables such as the increment of value *t*, the line's length, the number of lines/density which can either be fixed or modified by the user.

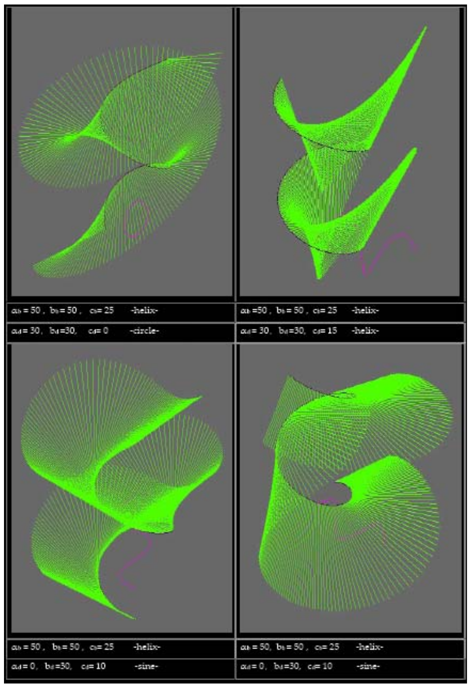

$a_b = 50$, $b_b = 50$, $c_b = 25$ -helix-
$a_d = 30$, $b_d = 30$, $c_d = 0$ -circle-
$a_b = 50$, $b_b = 50$, $c_b = 25$ -helix-
$a_d = 30$, $b_d = 30$, $c_d = 15$ -helix-
$a_b = 50$, $b_b = 50$, $c_b = 25$ -helix-
$a_d = 0$, $b_d = 30$, $c_d = 10$ -sine-
$a_b = 50$, $b_b = 50$, $c_b = 25$ -helix-
$a_d = 0$, $b_d = 30$, $c_d = 10$ -sine-

Fig. 2.24: Composed surfaces with equivalent values for a, b, c for the base curve and director curve.

2-2-3-2 Parametric Spirals[1]

It is a generative system that investigates a variety of spirals that exist in nature with different proportions and utilities. Spirals and helices can be described by the relation between one rotation and one or several directional arrays. To construct the program, it was necessary to study mathematical functions that generate spirals and consequently helices.

These functions had been translated into Auto-LISP code in order to generate helical structures based on helices, spirals and ellipses. Also some trigonometric notions were necessary for the introduction of the third dimension and respective variables.

(Fig. 2.25) shows Render of spirals made with Parametric spiral, an arithmetic spiral and a spiral made with a low quantity of segments. The number (1) corresponds to the number of turns; (2) distance between turns; (3) Extrusion ray; (4) Number of segments per turn – geometric resolution.

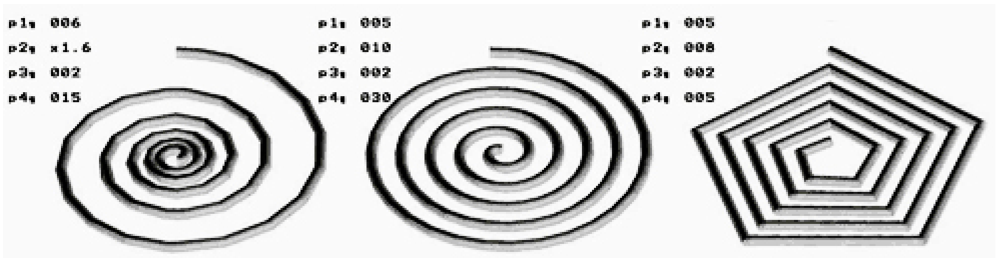


Fig. 2.25: Render of spirals made with a LISP language in Parametric spirals.

- **The Process**

The variety of structures generated by the program was achieved by the inclusion of numerous variables. The parametric control of those variables allows the adjustment of the development's direction, dimension, number of elements, proportions and rhythms involved in the creation of the structure.

[1] Couceiro, M.: *Architecture and Biological Analogies, Finding a Generative Design Process Based on Biological Rules*. Escola Tecnica Superior d´Arquitectura – Universidad Internacional de Catalunya, Spain, eCAADe 23 - session 13: generative systems, 2005.

(Fig. 2.26) shows how even with small changes in one or two parameters at a time, the difference between the results is considerable.

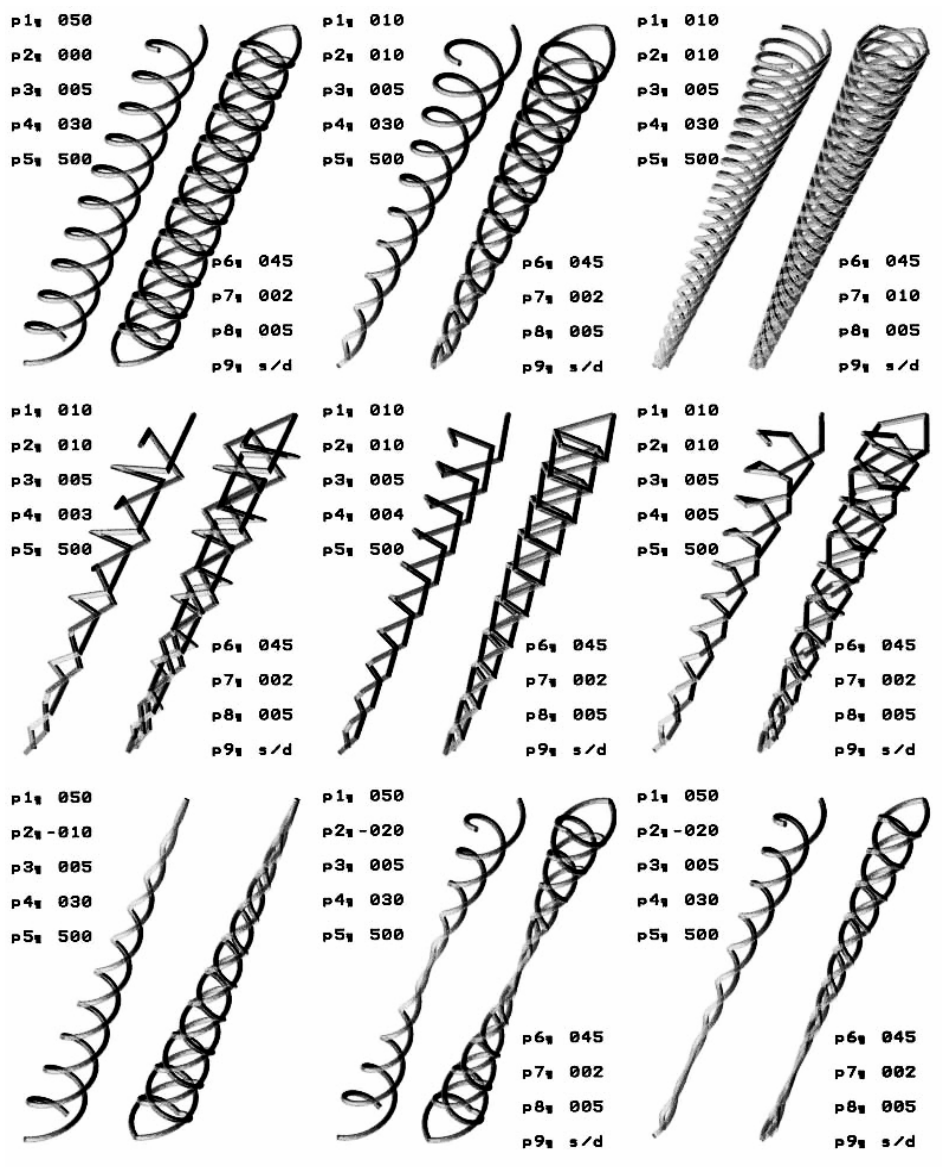


Fig. 2.26: Board with renders of helices made by the program with different variables applied. (1) Initial ray value to the variable; (2) Distance between turns; (3) Number of turns; (4) Number of segments per turn - geometric resolution; (5) Total helices height; (6) Angle with the z axis; (7) Number of helices; (8) Extrusion ray; (9) Single or double rotation[1].

[1] Ibid.

There is an intermediate function inside the code that is responsible for the translation of measures and the calculation of the variables to be applied in the main function.

Then the last function generates the helices reference points, based on processes of repetition, associated with recurrent sub functions, in a system of polar coordinates.

The use of this software can also be tried on itself. This means that the main mathematical function that generates the reference points to draw the axis of the helices was used to generate values to some of the variables of similar mathematical functions, resulting more complex spirals (Fig. 2.27).

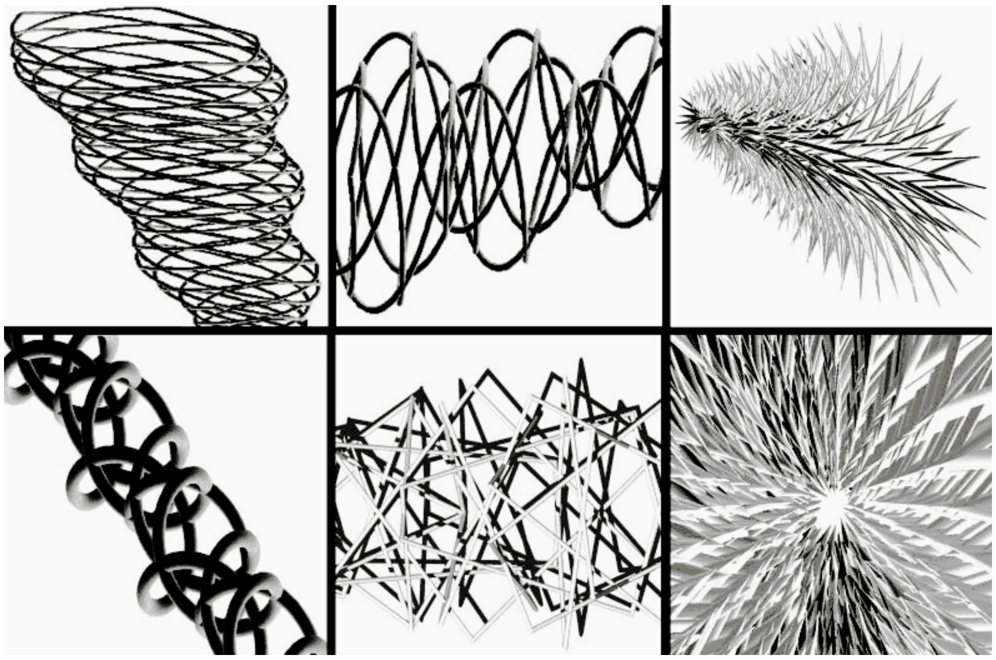


Fig. 2.27: Renders of some applications of parametric spiral generator.

2-3 Algorithmic Form Generation

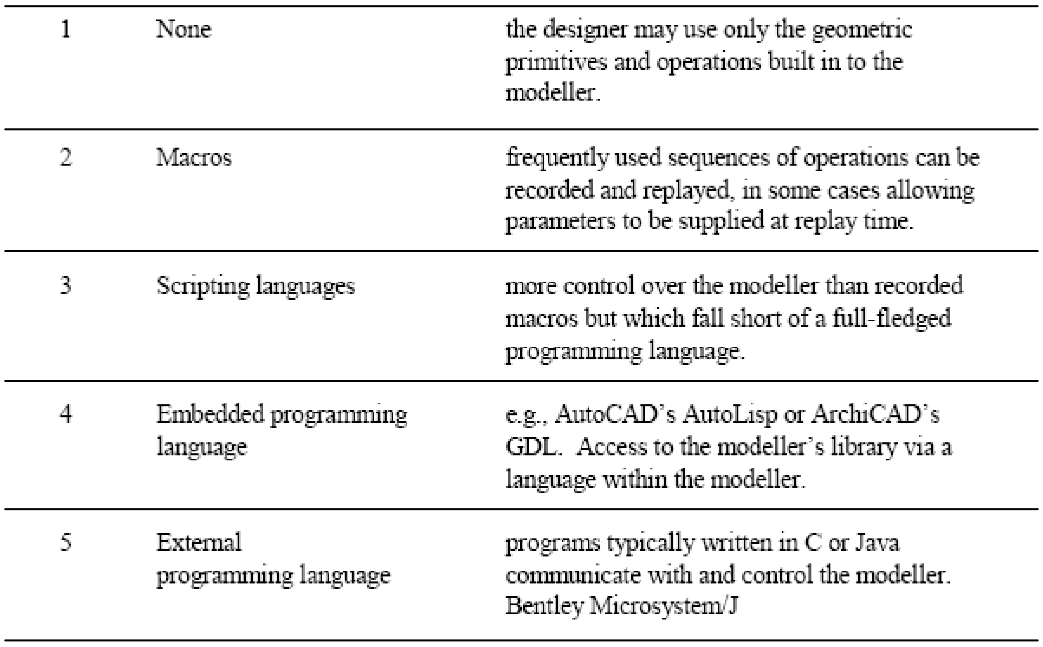
2-3-1 Description

Generating form algorithmically is writing mathematical rules in a computation medium whose execution results are two or three dimensional geometry.

Generally, creating a form using algorithms depends mainly, on two types of software, previously designed software for modeling forms, or writing an algorithm using programming languages (a small design program that executes forms)[1].

- **Types of Algorithmic Form Generation**

(Tab. 2.4) distinguishes five types for algorithmic form generation provided by CAD modelers. Most CAD modeling programs offer macro facilities or scripting languages.



1	None	the designer may use only the geometric primitives and operations built in to the modeller.
2	Macros	frequently used sequences of operations can be recorded and replayed, in some cases allowing parameters to be supplied at replay time.
3	Scripting languages	more control over the modeller than recorded macros but which fall short of a full-fledged programming language.
4	Embedded programming language	e.g., AutoCAD's AutoLisp or ArchiCAD's GDL. Access to the modeller's library via a language within the modeller.
5	External programming language	programs typically written in C or Java communicate with and control the modeller. Bentley Microsystem/J

Tab. 2.4: Five levels of support for algorithmic form generation.

[1] Gross, M.D.: *FormWriter: A Little Programming Language for Generating Three-Dimensional Form Algorithmically*. Computer Aided Architectural Design Futures 2001, Kluwer Academic Publishers, 2001.

i. None

Provide no support whatsoever for algorithmic generation: models must be constructed directly using the geometric primitives and operations provided on the CAD modeler's menus.

ii. Macros

Serves simple tasks well (such as repetitive window patterns or stairs), it is difficult to program more complex operations using only macros.

iii. Scripting Languages

Gained wide acceptance in other domains (JavaScript and Flash), provide considerably more power than macros but coding more sophisticated tasks and complex, requiring a specialist programmer.

iv. Embedded Programming Language

Like a scripting language, enables the programmer to control and command the modeler from an environment within the CAD program, and allows more powerful constructs than the typical scripting language.

Many CAD programs now include an embedded language, and advanced users of these CAD programs enthusiastically endorse their modeler’s scripting or embedded language.

AutoLisp is arguably the best known example. Although the underlying Lisp language is extremely elegant and powerful, Autodesk’s implementation was a weak one and the programming environment for developing AutoLisp routines is woefully inadequate by modern standards[1].

[1] Ibid. p.579-580.

Another example of an embedded programming language is ArchiCAD’s GDL, which provides access to the modeler's functionality through a BASIC-like language.

Although it provides this functionality, the choice of a BASIC programming style limits the language and renders it inelegant, Language design makes an enormous difference. GDL does enable the construction of parametric objects.

v. External Programming Language

Such as C or Java can be used to write complex form-generating algorithms but it requires more expertise than most designers are willing to commit to acquiring.

Some designers have resorted to using software such as Mathematica or MathCAD to generate three-dimensional surfaces.

If the designer wants to generate three dimensional forms algorithmically, one must decide between two main alternatives[1]:

- Macro facilities and scripting languages within CAD modelers are relatively easy to learn, but they inherently limit the programs one can write (and hence the forms one can generate).

- Full-fledged programming languages such as C and Java are powerful but they require more effort to learn, and generating 3D geometry also requires attention to many language features that have no direct bearing on form.

[1] Ibid. p.578.

2-3-2 Architectural Potentials

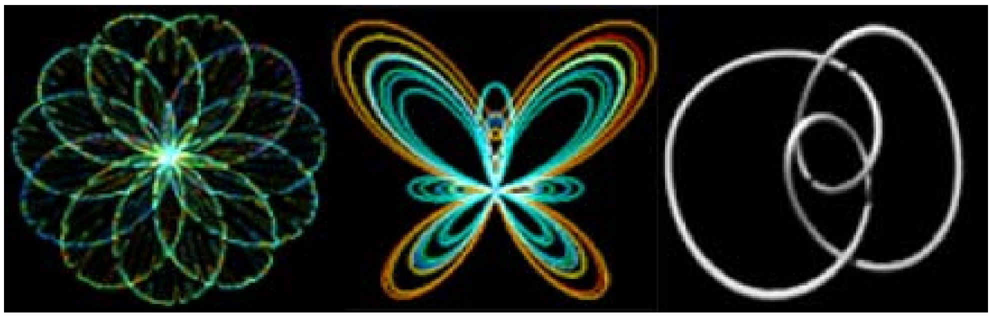
2-3-2-1 Generating Complex Forms

i- Curves

Algorithmic form generation can be used to generate a variety of curves useful for generating architectural floor plans, architectural ornamentation as well as generating geometric spaces for building domes or Greg Lynn and Frank Gehry's organic forms and spaces. (Tab. 2.5, Fig. 2.28, 2.29) shows samples of the curve types that can be generated using algorithms[1].

Super circle	Square	Lemniscate	Bicorn curve
Piecewise cubic	Archimedes	Deltoid	Glissette
Bezier	Spiral	Strophoid	Diamond curve
Butterfly curve	Sinusoidal	Nephroid	Kappa curve
Chrysanthemum	Spiral	2D Bow curve	Piriform curve
Cycloid	Coth spiral	Bow curve	Trisectrix of
Spline curves	Tanh spiral	Klein Cycloid	Maclaurin
Conic sections	Cornu Spiral	Krishna Anklets	Tractrix
Archimedes	Cayleys sextic	Mango Leaf	Folium curve
Spiral	Cissoid of Dioches	Snake Kolam	Baseball seam
Equiangular	Viviani	Cardioid	Eight knot
Spiral	Epicycloid	Parabola	Helix
Fermats Spiral	Concoid of	Spherical	Limacon
Hyperbolic	Nicromedes	Cardioid	Spherical
Spiral	Cissoid of	Astroid	Nephroid
Lituus Spiral	Dioches	Hyperbola	Freeths
Parabolic spiral	Viviani	Hypocycloid	Nephroid
	Epicycloid	Agnesi curve	Borromean rings

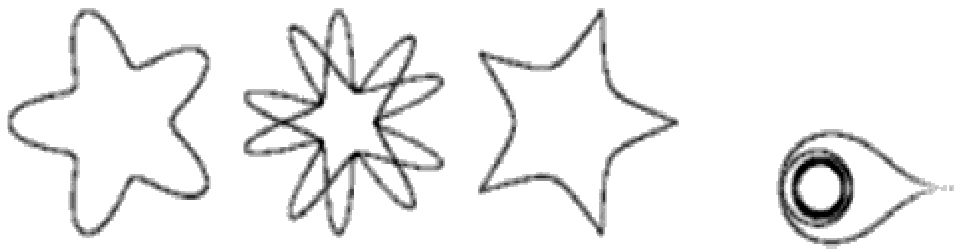
Tab. 2.5: Samples of curve types generated using algorithms.



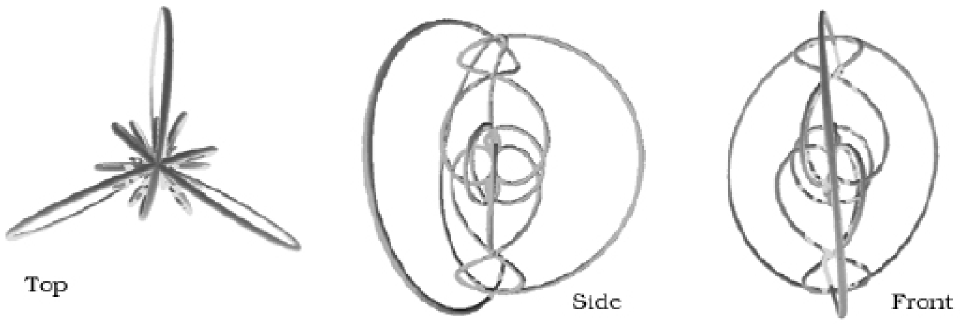
Chrysanthemum curve butterfly curve knot curve

Fig. 2.28: Rendered curve families generated using algorithms.

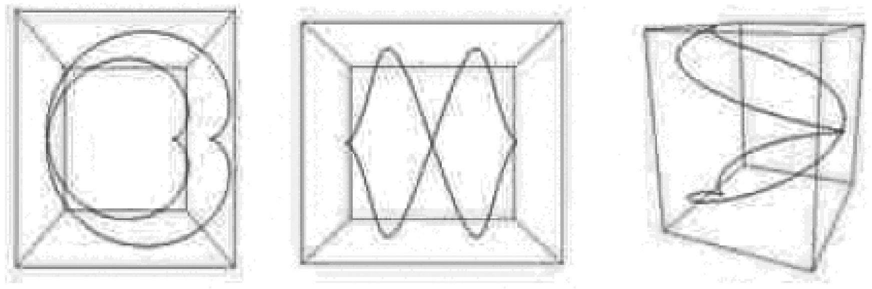
[1] Mohammadi, G.: *Geometric Shape Generator*. M.Sc. thesis proposal, University of Washington, 2004.



(a) Super shapes



(b) Knots



(c) Spherical cardioids

Fig. 2.29: More rendered curve families can be produced using algorithms[1].

[1] Ibid. p.20.

ii- Surfaces

Surfaces as well as curves can be generated using mathematical algorithms. (Tab. 2.6, Fig. 2.30) illustrates some of the surface categories that could be generated using algorithmic form generation[1].

Spherical Harmonics	Hearts	Twisted pipe
3D Super Shape	Mobius strip	Devil surface
Pseudo sphere	Pisod triaxial	Swallow
Steinmetz solid	Dini's surface	P1 atomic orbital
Bifolia surface	Tear drop	Ghost Plane
Twisted plan	Verrill surface	Folium
Kuen's surface	Cassini Ova	Bent Horns
Cross Hole	Glob teardrop	Tubey
Twisted Fano	Piriform	Catenoid minimal surface
Fano planes	Gerono lemniscat	Helicoids minimal surface
Slipper surface	Piriform	Catalan minimal surface
Tranguloid Trefoi	Cassini Oval	Henneburg minimal
Chladni plates	Glob teardrop	surface
Nose, Witch hat	Jet surface	Scherk minimal surface
Tangle, Strophoid surface	Superellipse	Bour minimal surface
Chair surface	Cymbelloid	Ennepers minimal surface
Fish, Cresent	Bezier surfaces	Richmond minimal surface
Sea Shells	Spline surfaces	Handkerchief, Kidney
Steiner look-a-like	Triaxial teardrop	Monkey saddle
Cross Cap	Whitney umbrella	Pillow shape
Steiner surface	Lemniscape	Snail surface
Maeder's Owl	Tractrix	Kinky torus
Twisted Triaxia Hunt	Pseudocatenoi	Kusner Schmitt
surface	Twisted heart	McMullen K3 mode
Stiletto surface	Torus/Supertoroid	Weird surface Tri-Trumpet
Mitre surface	Triaxial Tritorus	Gerhard Miehlich
Blob, Nodal cubic	Elliptic Torus	Kampyle of Eudoxus
Boy surface	Gumdrop Torus	Barth sextic
Apple shape	Saddle Torus Bohemian	Cayley surface
Klein Cycloid	Dome	Tooth

Tab. 2.6: Surfaces categories generated using algorithmic form generation.

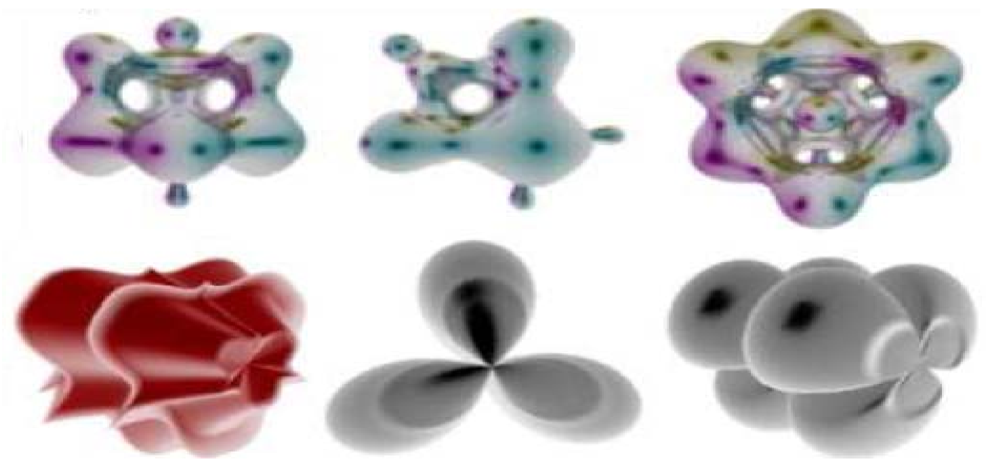


Fig. 2.30: Examples of Blobs and super shapes surfaces generated using algorithms.

[1] Ibid. p.20.

2-3-2-2 Analysis of Style

The following research model illustrates experiment in algorithmic form generation in Autolisp, AutoCAD programming language, to generate "Corbu" like prototypes.

- **Autolisp Routine for Le Corbusier Styles[1].**

In Le Corbusier's own summary of his main architectural elements he identified his buildings to be made up of the five points of Architecture; Pilotis, Roof Garden, Free plan, Ribbon Windows, and Free façade (Fig. 2.31). The Pilotis raised the building off the ground into the air and the space underneath used for parking cars, road or gardens. Space lost was replaced on the top by a roof garden. The Roof Garden was a space open to the sky, containing greenery with view all round. Free planning was possible since the frame carried the weight partitions.

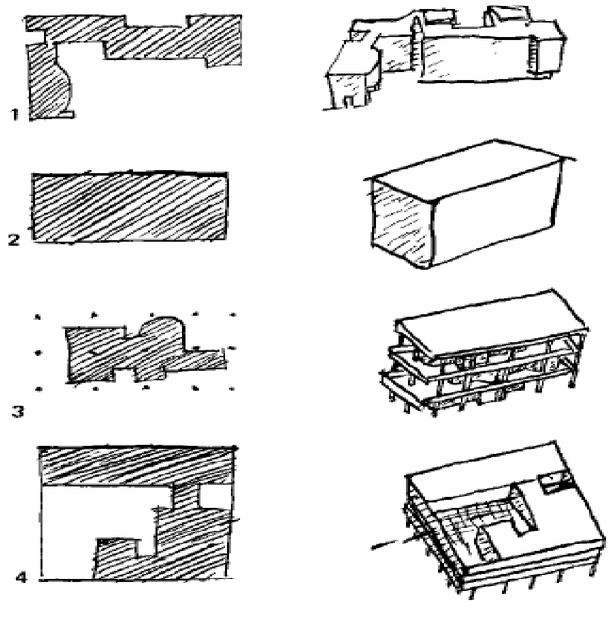


Fig. 2.31: Illustrations for Le Corbusier's main points of Architecture[2].

[1] Abimbola, O. A.: *Exploring Algorithms as Form Determinants in Design*. The University of Oklahoma, USA, the 3rd International Space Syntax Symposium, Atlanta, 2001.
[2] Boesiger, W.: *Le Corbusier*. Ingoprint, Barcelona, 1992.

Six of Le Corbusier's buildings were selected (Tab. 2.7) to develop an algorithm for generating typical styles. These buildings were chosen because of their similarities, size, simplicity of language of design and relationship to the five points of architecture, which is the starting point of the design algorithm.

Project	Building Form	Major Elements	Spaces
Citrohan House 1920. (Figure 2)	Rectilinear and cubical in configuration. The building is subdivided into two cubical forms.	Double height space in living room External staircase Interior spiral stairs Zoning-Living, cooking, dining and sleeping.	Living, gallery, roof terrace, bedroom and kitchen.
Villa at Vaucresson 1922. (Figure 3)	Double cubic form, building form based on relationship between two masses. Contrast between smaller vertical and larger horizontal elements	Sloped site Entry is placed between two masses Clearly defining major and minor elements. Detached staircase Main axis parallel with road.	Living, dining and bedrooms.
Houses for Workers 1924. (Figure 4)	Cube with triangular Upper floor slab	Double height living room Three activity zones Diagonal reinforced with straight flight of stairs. Minimum accommodation requirements	Living, dining, Bedroom and kitchen.
Weissenhof house1924. (Figure 5)	Rectilinear configuration. Building raised on pilotis	Main staircase is enclosed Roof terrace Curved elements control movement Pilotis imply garage is underneath building and living room on 1st floor. 3 columns in transverse direction 5 columns in longitudinal direction. Double height living spaces	Roof terrace, living, bedroom, dining and kitchen.
Villa at Garches for Michael Stein, 1927 (Figure 6)	Large scale domino structure. Mass around the reinforced concrete frame.	5 columns in longitudinal and transverse direction. 5,2.5,5,2.5,5 an ABABA relationship longitudinally. 1.25,4,3,4,1.25 an ABCAB relationship transversely. Two apsidal staircase Convex and concave walls define spaces. Free forms. Apsed hammerhead forms.	Living, dining, Bedrooms, gallery and roof terraces

Tab. 2.7: Summary for major elements from five of Le Corbusier's buildings[1].

[1] Abimbola O. A.: *Exploring Algorithms as Form Determinants in Design*. The University of Oklahoma, USA, the 3rd International Space Syntax Symposium, Atlanta, 2001.op.cit.

- **Program Rules**

The program rules were based on the main elements of Le Corbusier's architectural style under the following main categories:

1. Orthogonal Cage determination
2. Circulation
3. External walls
4. Main curved elements of interior

These categories were further elaborated into rules specifying for example: points, vectors, polygons and other graphic token that could be interpreted in Autolisp[1].

AutoCAD's built in programming language Autolisp was chosen based on AutoCAD's adaptability and popularity. Autolisp is derived form common lisp and is open for customization to specific needs. Lisp presents information in form of lists. Lisp is known to be the most extensive of computer languages, it has about 200 to 300 built in functions and the programmer can also create their own functions. Lisp functions are used to express data.

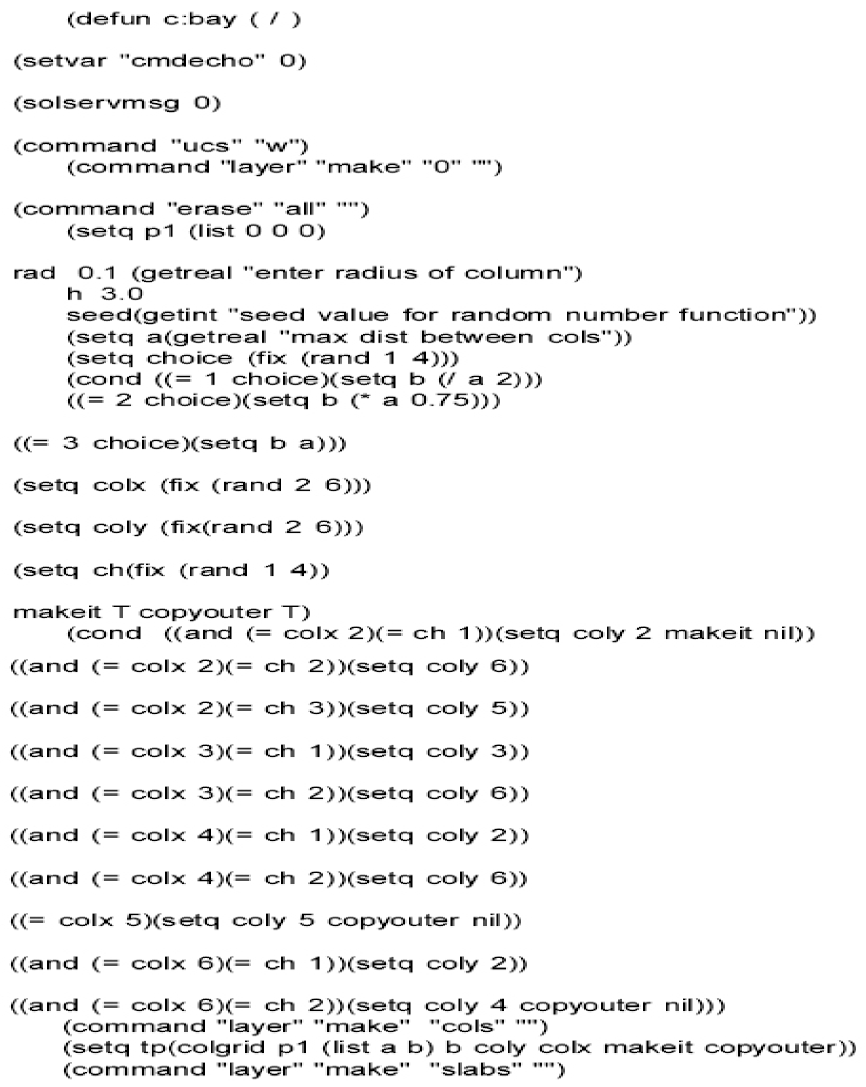
In the program separate functions were written for main elements like the column grid, circulation elements, terraces, curved screens, external wall placement etc. The functions were given separate arguments based on the rules and a main function evaluates all these separate functions.

The use of randomness was incorporated such that the program determines the evolution of the design thus exploring architecture as self generated. The Autolisp routine is loaded from the command prompt in an AutoCAD drawing file, which then prompts the user for a random number function. The user can input a random number between 0 and 32567, then the program by itself selects a column grid system based on the program rules.

[1] Ibid. p.24.7.

Upon determining the column grid system it determines the number of slabs, types of external faces, elements of interior and then the curved screen.

(Fig. 2.32) shows the main part of the 16 pages Autolisp Routine. This routine is based on the arguments and rules which were derived from the following categories: Orthogonal Cage determination, Circulation, External walls and main curved elements of interior.



```
    (defun c:bay ( / )

(setvar "cmdecho" 0)

(solservmsg 0)

(command "ucs" "w")
    (command "layer" "make" "0" "")

(command "erase" "all" "")
    (setq p1 (list 0 0 0)

rad  0.1 (getreal "enter radius of column")
    h  3.0
    seed(getint "seed value for random number function"))
    (setq a(getreal "max dist between cols"))
    (setq choice (fix (rand 1 4)))
    (cond ((= 1 choice)(setq b (/ a 2)))
    ((= 2 choice)(setq b (* a 0.75)))

((= 3 choice)(setq b a)))

(setq colx (fix (rand 2 6)))

(setq coly (fix(rand 2 6)))

(setq ch(fix (rand 1 4))

makeit T copyouter T)
    (cond  ((and (= colx 2)(= ch 1))(setq coly 2 makeit nil))
((and (= colx 2)(= ch 2))(setq coly 6))

((and (= colx 2)(= ch 3))(setq coly 5))

((and (= colx 3)(= ch 1))(setq coly 3))

((and (= colx 3)(= ch 2))(setq coly 6))

((and (= colx 4)(= ch 1))(setq coly 2))

((and (= colx 4)(= ch 2))(setq coly 6))

((= colx 5)(setq coly 5 copyouter nil))

((and (= colx 6)(= ch 1))(setq coly 2))

((and (= colx 6)(= ch 2))(setq coly 4 copyouter nil)))
    (command "layer" "make"  "cols" "")
    (setq tp(colgrid p1 (list a b) b coly colx makeit copyouter))
    (command "layer" "make"  "slabs" "")
```

Fig. 2.32: Main part of Autolisp Routine for Le Corbusier's style[1].

[1] Ibid. p.24.9-24.10.

The Autolisp routine generates a wide variety of Corbusier building types and supports the idea of generating complex outcomes from simple rules. The designer also received some unexpected design results. (Fig. 2.33) illustrates some of the "Corbu" like prototypes generated.

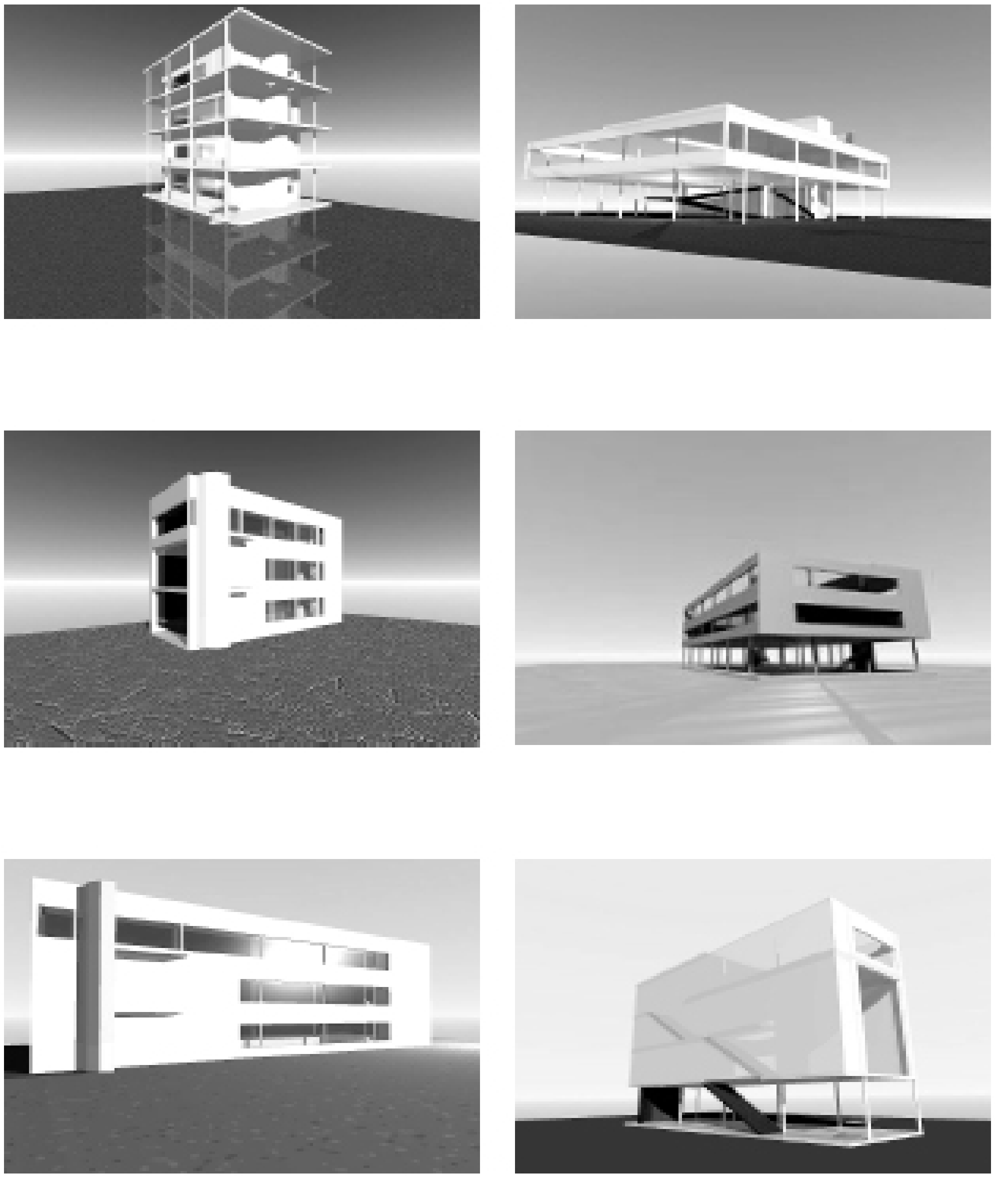


Fig. 2.33: "Corbu" like prototypes generated[1].

[1] Ibid. p.24.9-24.10.

2-3-3 Algorithmic Form Generative Systems

Papert and Feurzeig in 1967 created the first version of Logo with a team from Bolt, Beranek and Newman[1]. Logo is easy to use and learn. It has been developed over the past 36 years.

Designers can explore formal design ideas by making an algorithm for shape generation in Logo. The Logo programming environment uses a Turtle (Fig. 2.34a) that can be directed by commands typed at the computer. For example, the command [forward 50] causes the turtle to move forward in a straight line 50 "turtle steps". The command [right 90] rotates the turtle 90 degrees clockwise without moving any step. Then [forward 50] causes it to go forward 50 steps in the new direction. The turtle generates a square shape following the sequential commands (Fig. 2.34b). This square can also be used as a procedure in a later instruction. Repeating the commands, while rotating 10 degree each turn the turtle generates a flower-like shape (Fig. 2.34c).

This sequence of commands shows one way of generating shape configurations with simple operations. This simple process of Logo Programming is used in various systems for design like Design By Number and FormWriter. These two projects provide a symbolic language with which users can write and debug formal descriptions of geometric objects, which are then rendered visually by the program.

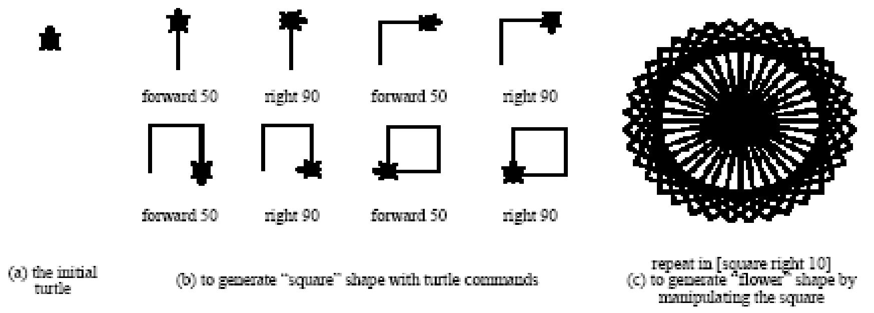
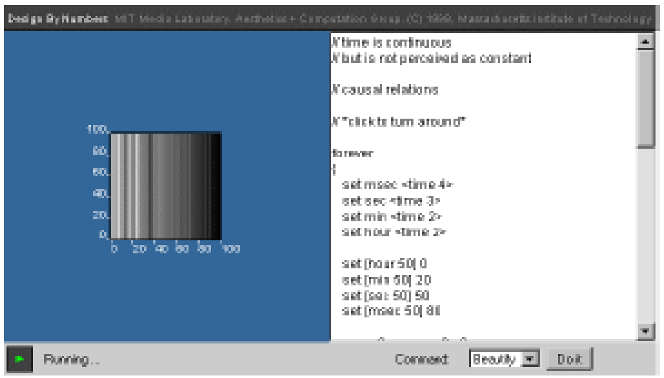

forward 50
right 90
forward 50
right 90
forward 50
right 90
forward 50
right 90
(a) the initial turtle
(b) to generate "square" shape with turtle commands
repeat in [square right 10]
(c) to generate "flower" shape by manipulating the square

Fig. 2.34: The turtle commands to generate shape configuration in Logo Programming Environment. Source: http://el.media.mit.edu/logo-foundation/.

[1] Kwon, D.Y.: *ArchiDNA: A Generative System for Shape Configuration*. Master of Science in Architecture, University of Washington, 2003.

2-3-3-1 Design By Number

Maeda[1] developed a two-dimensional programming system called Design by Number. He implemented this system as a tool for teaching 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. (Fig. 2.35) shows the interface of the system. The interface allows a designer to write simple code and use the code to produce visual 2D images. (Fig. 2.36) shows the various results produced using different user-programmed algorithms.



```
// time is continuous
// but is not perceived as constant

// causal relations

// "click to turn around"

forever
{
  set msec <time 4>
  set sec <time 3>
  set min <time 2>
  set hour <time 2>

  set [hour 50] 0
  set [min 60] 20
  set [sec 50] 50
  set [msec 50] 80
```

Fig. 2.35: Snapshot of Design by Number Interface. Source: http://dbn.media.mit.edu/.

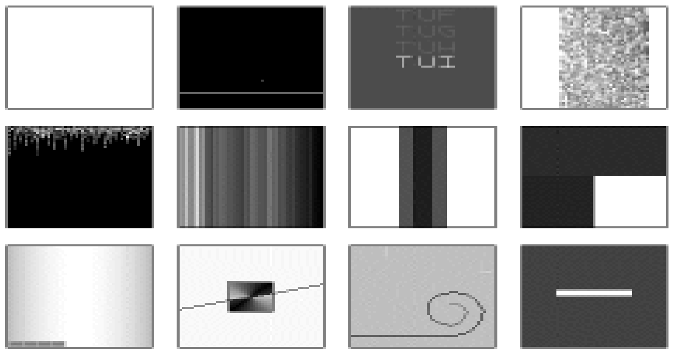


Fig. 2.36: Various Results of Design by Number. Source: http://dbn.media.mit.edu/.

[1] Maeda, J.: *Design by Number*. The MIT Press. Cambridge, MA, 1999.

2-3-3-2 FormWriter

Gross[1] 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. This programming language enables architects to explore 3D geometry form by simple programming.

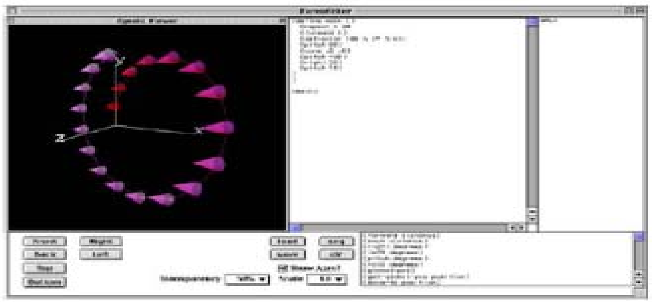


Fig. 2.37: Snapshot of FormWriter Interface. Source: Gross, 2001.

The interface consists of an editor window for writing code and a 3D space with browsing controls (Fig. 2.37). A designer writes simple commands on the editor window. It positions geometric elements in a 3D space. The designer views the generated forms through the 3D space with browsing controls. FormWriter can be used to generate interesting Islamic architectural forms (such as star rib frame and muqarnas dome) with no previous programming experience (Fig. 2.38).

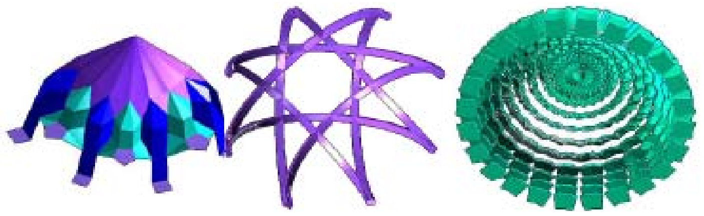


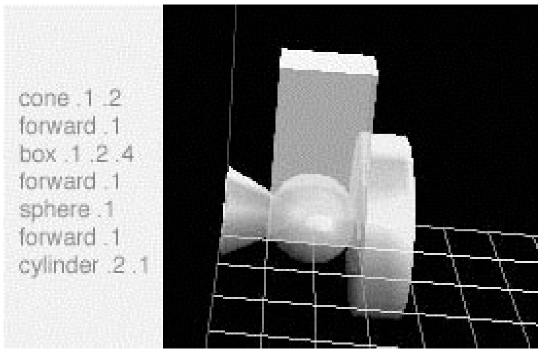
Fig. 2.38: Models of Islamic structures by students using the FormWriter. Source: Gross, 2001.

[1] Gross, M.D.: *FormWriter: A Little Programming Language for Generating Three-Dimensional Form Algorithmically*. Computer Aided Architectural Design Futures 2001, Kluwer Academic Publishers, 2001.op.cit.

- **FormWriter Process**

i- Writing Commands[1]

Form writer enables writing codes directly, without defining any procedures. (Fig. 2.39) shows simple FormWriter steps along with the lines of code that generated it.

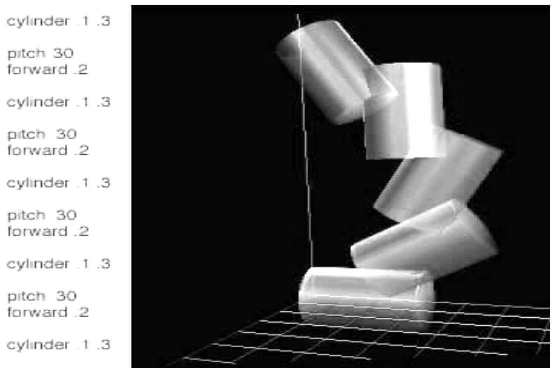


```
cone .1 .2
forward .1
box .1 .2 .4
forward .1
sphere .1
forward .1
cylinder .2 .1
```

Fig. 2.39: Generating simple forms directly.

FormWriter's primitive procedures to generate geometry (triangle, cone, box, sphere and cylinder) take dimensions as parameters, the forms are positioned by moving the 3D "flying turtle" forward between the primitive geometry generating procedure call.

(Fig. 2.40) shows how the 3D (flying) turtle is used to position and orient five cylinders. The flying turtle can move forward and back, and turn (right and left), pitch (up and down), and roll (side to side).



```
cylinder .1 .3
pitch 30
forward .2
cylinder .1 .3
pitch 30
forward .2
cylinder .1 .3
pitch 30
forward .2
cylinder .1 .3
pitch 30
forward .2
cylinder .1 .3
```

Fig. 2.40: Positioning cylinders in space using (pitch and forward).

FormWriter inserts each geometric primitive at the current position and orientation of the flying turtle, so as the turtle moves forward (2 units) and pitches up (30 degrees), between calls to the "cylinder" primitive procedure, each cylinder is translated and rotated in space.

[1] Ibid. p.581-582.

ii- Defining Procedure[1]

FormWriter employs the same editor window for defining procedures as for executing simple commands directly. Each of the forms in (Fig. 2.41) was produced by iterative calls to the user-defined procedure as shown in (Fig. 2.42). Adding first a turn, then a roll, and finally a pitch instruction. Row of boxes (Fig. 2.41 a), turning row (Fig. 2.41 b), twisting turning row (Fig. 2.41 c) and helix (Fig. 2.41 d).

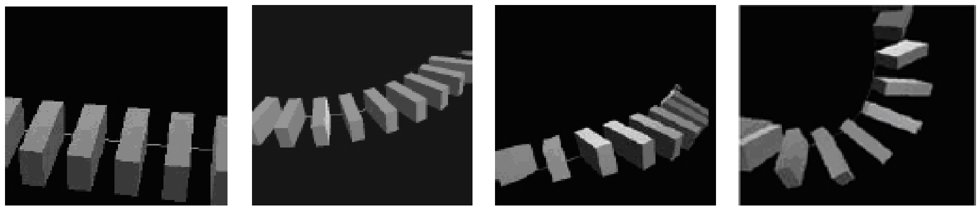


Fig. 2.41: (a) A row of boxes; (b) turning boxes; (c) twisting boxes; (d) boxes helix.

```
to one_box()          to one_box()          to one_box()          to one_box()
    box(.4,.1,.2)         box(.4,.1,.2)         box(.4,.1,.2)         box(.4,.1,.2)
    forward (.2)          forward (.2)          forward (.2)          forward (.2)
end                       right(10)             right(10)             right(20)
                      end                       roll(10)              roll(20)
                                            end                       pitch(20)
                                                                  end
```

Fig. 2.42: Recursive codes for the generated boxes.

(Fig. 2.43) shows variations produced by a similar program. Generated triangles are written with an input in (Fig. 2.44).

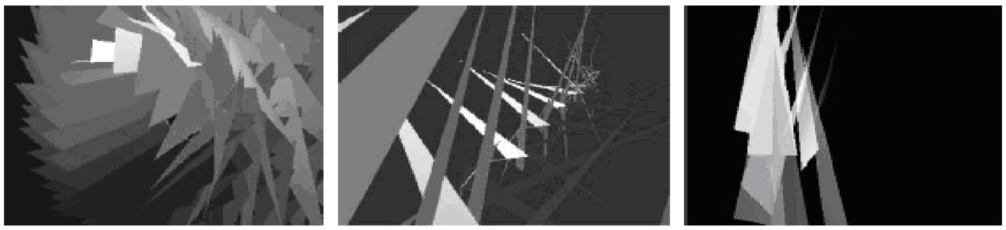


Fig. 2.43: Triangle variations: Orchid, Spiny Tree, and Cathedral.

```
TO TRIMANY (N)                            TO TRI (N)
 TRI(N)                                   COLOR (.6,.1,.7)
 FORWARD((N/ 20) )                         TRIANGLE(0, 0, N, N, 0, 0)
 ROLL(20)                                  COLOR(.1,.7,.3)
 PITCH(10)                                 TRIANGLE(0, N, N, N, 0, 0)
 if (N> 0)   then TRIMANY((N- 1) )        end
end
```

Fig. 2.44: Recursive codes for the generated triangles.

[1] Ibid. p.583-584.

2-3-3-3 Processing[1]

Processing is a programming environment generating interactive 2D and 3D images. The idea is to support easy-to-use and simplified codes so that users can understand the concept of programming to create and interact with the images.

Using Processing, a designer generates interesting graphic images (Fig. 2.45). Processing supports simplified codes that run through general Java applets.

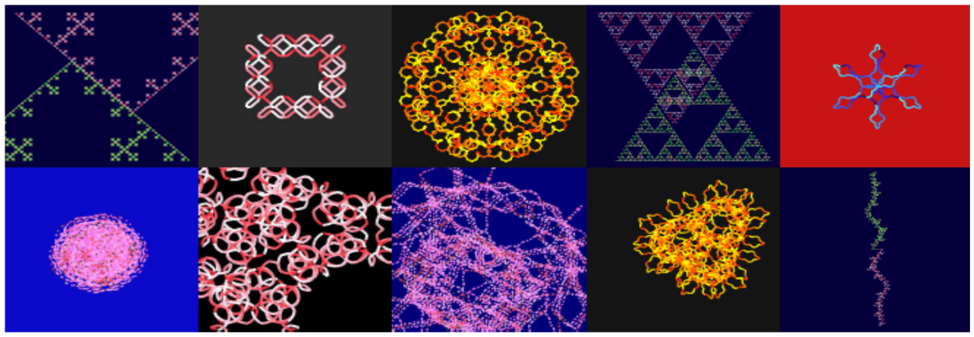
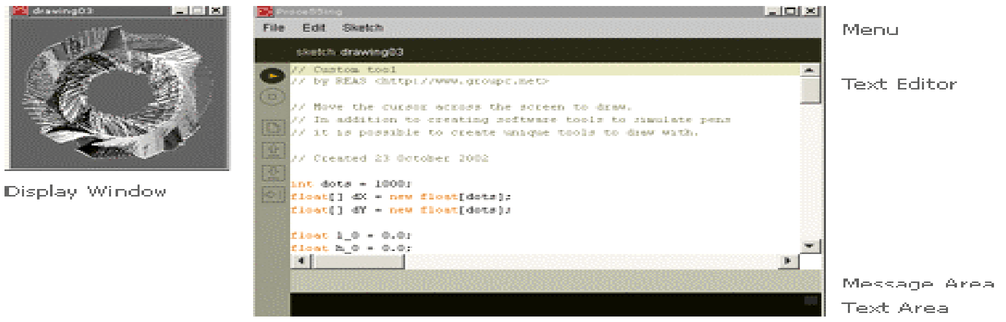


Fig. 2.45: Example of shapes generated with processing[2].

A designer writes a program in the text editor window and sees the visual result through the Java applet window. (Fig. 2.46) shows the working environment.



```
// Custom tool
// by REAS <http://www.groupc.net>

// Move the cursor across the screen to draw.
// In addition to creating software tools to simulate pens
// it is possible to create unique tools to draw with.

// Created 23 October 2002

int dots = 1000;
float[] dX = new float[dots];
float[] dY = new float[dots];

float l_0 = 0.0;
float h_0 = 0.0;
```

Fig. 2.46: Snapshot of Processing Working Environment. Source: http://www.proce55ing.net/

[1] Kwon, D.Y.: *ArchiDNA: A Generative System for Shape Configuration*. Master of Science in Architecture, University of Washington, op.cit.
[2] Mohammadi, G.: *Geometric Shape Generator*. M.Sc. thesis, University of Washington, op.cit.

2-3-3-4 **ArchiDNA** (a generative system inspired by Eisenman)[1]

ArchiDNA specifies design style and form generating rules to produce automatically spatial configurations. ArchiDNA is inspired by Peter Eisenman's design of Biocentrum, an example of form generation from abstract design concepts. Eisenman developed the building form with the concepts of DNA (Fig. 2.47).

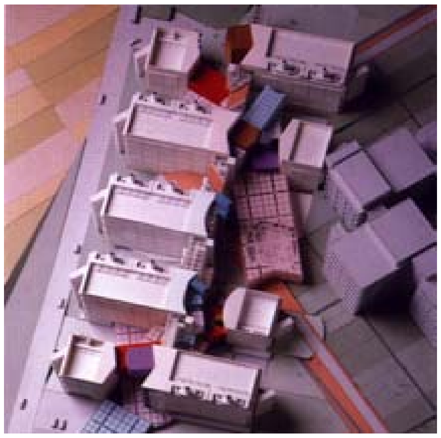
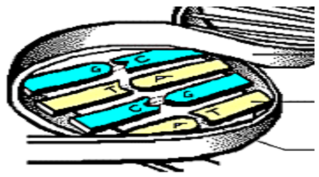


Fig. 2.47: Biocentrum (Eisenman 1986).

A DNA chain is composed of four initial shapes A, T, C, and G,2 interlocking pairs (Fig. 2.48 a, 2.48 b). Observing Eisenman's design, it is found that his principles of form generation are (1): replication of the source forms, (2) rotation of the generative form, (3) rescaling of the generative form to fit the width of the selected form[2] (Fig. 2.49).



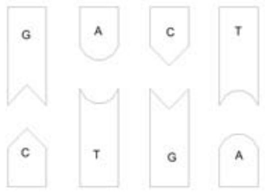


Fig. 2.48 (a): DNA Structure. (b): A, T, C, G.

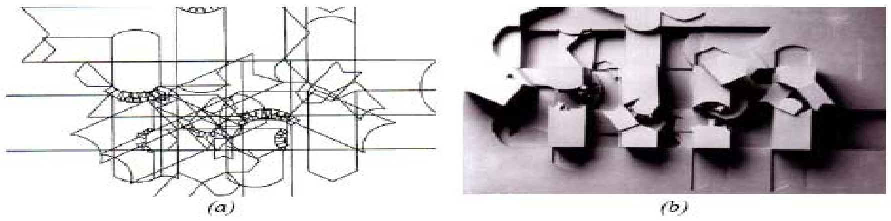


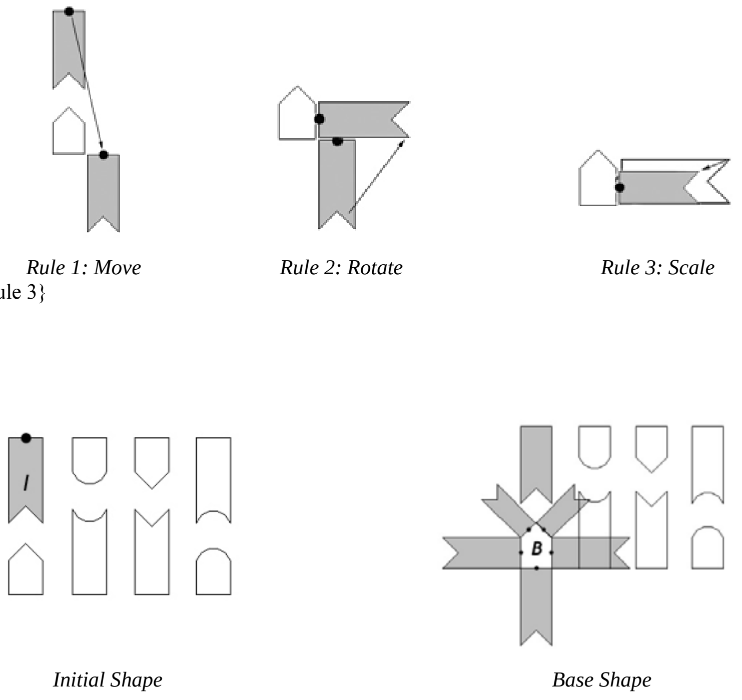
Fig. 2.49: Drawings for Biocentrum,Eisenman,1996 (a) Plan (b) Axonometric view.

[1] Kwon, D.Y.: *ArchiDNA: A Generative System for Shape Configuration*. Master of Science in Architecture, University of Washington, op.cit.

[2]Eisenman, P.: *Peter Eisenman Diagram Diaries*. Universe Publishing, New York, p. 27-43, 1999.

i- 2D Operations

ArchiDNA system, defines the form generating with five elements (S, L, G, B, I). S is a set of shape rules of the form [A → B] that specifies how a shape A can be transformed to create B. S are three parametric rules: Rule 1 (move), Rule 2 (rotate) and Rule 3 (scale). L labels the first edge. G is a set of parameters that assign values to transformation rules - the width and angle of each shape. B is the base shape to which the rule and source shapes apply to start a computation. I is an initial shape to act as a source for replication and as the base shape for orientation (Fig. 2.50).



S = {Rule 1, Rule 2, Rule 3}
L = {•}
G={g1, g2, g3}
I={Initial shape}
B={base shape}

Fig. 2.50: The rules used to manipulate the defined style[1].

ii- 2D Generation

ArchiDNA generate 2D drawings similar to Eisenman's plans for the Biocentrum building. We started with eight shapes (2 copies of each of the four DNA shape elements). The first step

[1] Kwon. D.Y., Gross. M.D. and Yi-Luen Do, E.: *ARCHIDNA: A Generative System for Shape Configuration*. Design Machine Group, University of Washington, USA, 2003.

is to apply the shape G (ribbon) as the applier-shape to the base-shape C (pentagon). ArchiDNA goes to work. It copies and attaches the applier-shape G to every edge of the base-shape C. (Fig. 2.51 a, 2.51 b) demonstrate 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. Results shown in (Fig. 2.52) and derivations shown in (Fig. 2.53, 2.54).

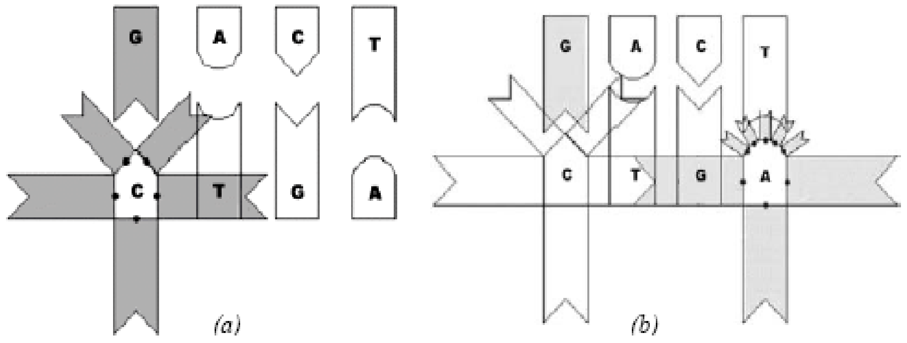


Fig. 2.51: (a) Applying the applier-shape G (ribbon) to the base-shape C (pentagon)
(b) Applying the applier-shape G to the base-shape A (arch).

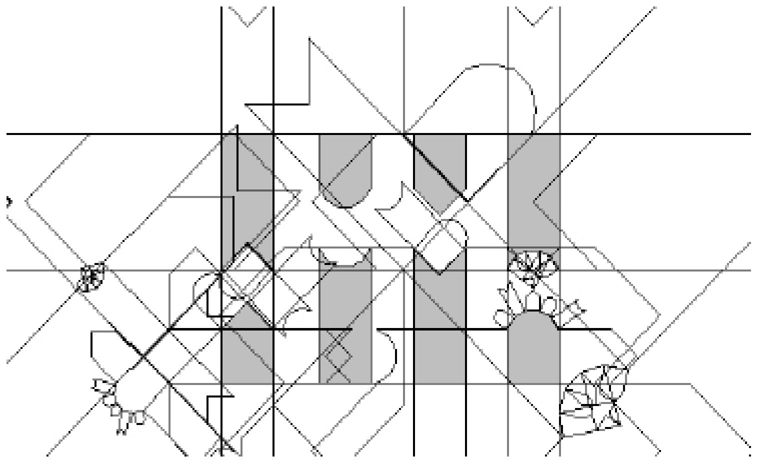


Fig. 2.52: 2D Eisenman-like drawing generated in ArchiDNA.

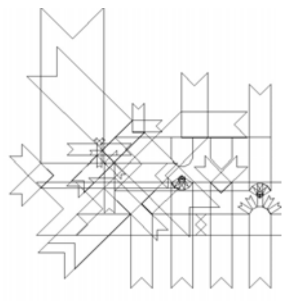


Fig. 2.53: Derivation 1 produced by ArchiDNA.

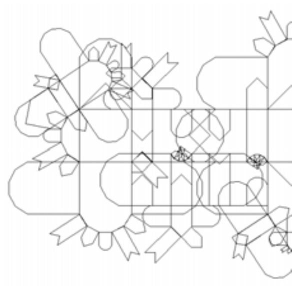


Fig. 2.54: Derivation 2 produced by ArchiDNA.

iii- 3D Operations

The 2D shapes can be converted to three dimensions by automatically assigning heights. Each of the eight shapes is extruded to a certain height, a function of its area. It appeared that a certain 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. (Fig. 2.55 a) 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. 2.55 b, 2.55 c). On the other hand, the height of shape B (ribbon) is fixed with a user-defined value (Fig. 2.55 d).

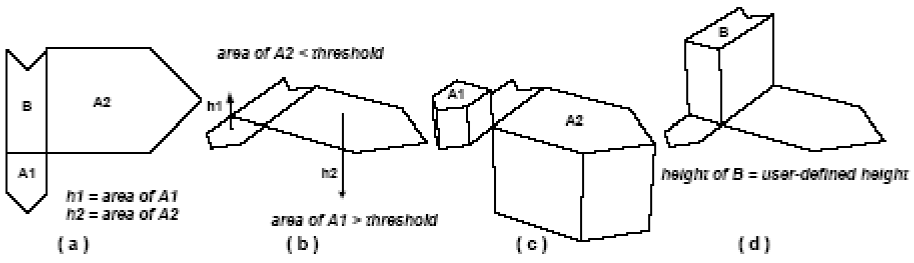


Fig. 2.55: 3D form generation in ArchiDNA (a) Calculating the area of two applied shape A1&A2 and assigning heights (b) Comparing areas with a threshold and deciding up and down (c) Extrusion of small shape A1 upward and extrusion of large shape A2 downward (d) Extrusion of shape B with a user-defined height.

iv- 3D Generation

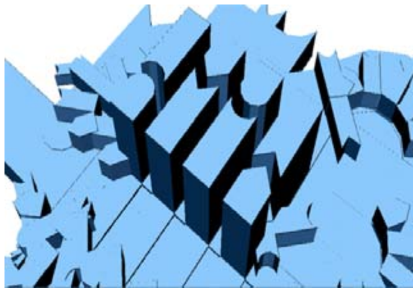


Fig.2.56: 3D Generated in ArchiDNA.

ArchiDNA automatically generates 3D objects by extruding the 2D drawings and exporting a 3D VRML file. (Fig.2.56) shows a 3D 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 function of their area.
ArchiDNA not only generates interesting 2D shapes but also extends to 3D massing and can translate the result into a 3D VRML format (Fig. 2.57) or CAD systems (Fig. 2.58).

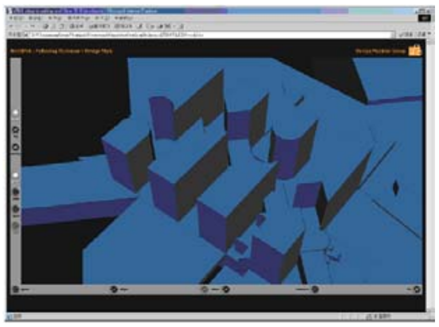


Fig. 2.57: ArchiDNA 3D Model in VRML Viewer, Cortona.

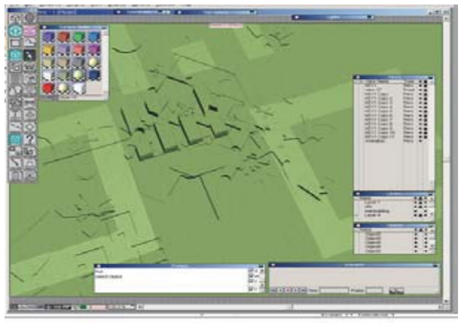
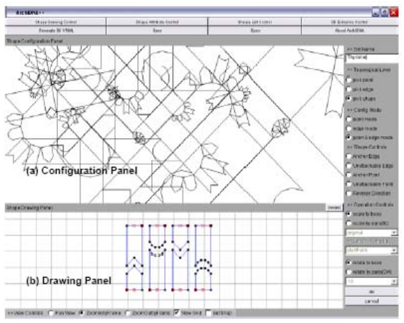


Fig. 2.58: ArchiDNA 3D Model in Modeling System, FormZ.

v- ArchiDNA Interface



ArchiDNA has an easy-to-use interface (Fig. 2.59). Designers can draw initial shapes and then select any shape to apply the algorithmic shape generations. All newly generated shapes can be selected as base shape to apply shape generation as well.

Fig. 2.59: Snapshot of ArchiDNA Interface.

Summary of chapter two:

Chapter two discussed three well-established themes in computer generated design. These form generative systems based on conventional mathematical principles like: Shape Grammars, Parametric Variations, and Algorithmic Form Generation. Shape Grammars as a tool used to generate orthogonal conventional forms using basic shape algebra and formal logic, and Parametric Variations as a tool used to generate complex three dimensional curves and folding surfaces using trigonometry and parameterized functions, and Algorithmic Form Generation as a tool used to generate and direct manipulate two dimensional and three dimensional forms using macro facilities and scripting languages or full fledged programming languages.

Chapter 3
Chaos Based Mathematical Generative Systems

Chapter 3: Chaos Based Mathematical Generative Systems

In the last years of the 19th century and in the beginning of the 20th, century a group of mathematicians led by Koch, Peano, Cesàro, Hilbert, Julia, Pointcaré, etc. started the study of the possibilities of new geometries, clearly different from the shapes and basic principles used in conventional mathematics, and opposed in its conception to the Euclidean geometry, preponderant until this moment.

Simultaneously to these precursors of modern geometry appears a new way of understanding and conceiving art, radically different from all the artistic tendencies developed until now, based on a new branch of mathematics known by chaos mathematics.

The Chaotic mathematical processes are not random, they follow rules, but even very simple rules can produce extreme complexity. Chaos mathematics leads to new geometries, these geometries cause a revolution in the scientific world generating an exciting family of images and forms whose basic generation elements are not Euclidean objects. Mathematics of chaos provides the tools for displaying such phenomenon.

In this method of creation, forms do not born in the designer's mind but they are generated by chaos in a computational medium. The designers only select, evaluate and transform that form.

This chapter will discuss two examples of these chaotic phenomena and studies their nature, mathematical concepts and architectural potentials. With the purpose to use them as tools of inspiration (inspiration triggers) or form generators for the modeling process, especially in the earlier stages of form finding. These tools are: Fractals and Spirolaterals.

3-1 Fractals

3-1-1 Introduction

3-1-1-1 Definition

The term fractal comes from the Latin word fractus, which means broken and irregular. Its own name indicates that it is a geometry that is specially thought for irregular objects -a geometry of nature- in contrast with the regular geometry created by men. A fractal is an object or quantity that displays self-similarity on all scales with non-integer dimensions. The object need not exhibit exactly the same structure at all scales, but the same “type” of structures must appear on all scales[1].

3-1-1-2 Outlines

The first confirmation of Chaos existence was done by Edward Lorenz in 1963. Lorenz was carrying computer research on weather. He wanted to clarify why there were discrepancies between the weather forecasted and the real one. He made a mathematical model of atmosphere composed of 12 equations. The system of equations was being solved by computer. Accidentally, to speed up the calculations, he introduced the intermediate values, which he got in a previous simulation and rounded off the values from 6 to 3 places after decimal point.

During the simulation of two months of weather this initially small difference of results became as big as the very signal from the beginning. In such a case if the real atmosphere behaves in the same way, it is impossible to forecast the weather months ahead. Even the most powerful computers won’t be of help here, as small mistakes would be growing to get big[2].

[1] Weisstein, E.W.: *Concise Encyclopedia of Mathematics*, CD-ROM edition 1.0, Chapman & Hall/CRCnetBASE, 1999.

[2] Jadwiga C.Z.: *Chaos, Databases and Fractal Dimension of Regional Architecture*. PhD in Architecture, Faculty of Architecture TU of Bialystok, Poland, eCAADe conference proceedings, Paris, France 24-26 September 1998.

The effect of sensitivity to the initial conditions was called ‘the butterfly effect’. (It comes from Lorenz’s publication ‘Can a flap of a butterfly wings in Brazil stir up a tornado in Texas?’). It means that due to chaos, even the smallest events can have great consequences. This Unpredictable behavior of deterministic systems has been called chaos[1].

In the 70’s some scientists in the USA and in Europe started to find their way through the chaos. They were dealing with different spheres of science: mathematics, physics, biology, chemistry, physiology, ecology, economy. In the next 10 years’ time the term ‘Chaos’ has become generally known in art. In 1975 Benoit Mandelbrot called them (fractals). Fractal geometry that described fractal objects was also his invention.

3-1-1-3 The Mathematical Concept

The mathematical nature of fractals based on chaotic mathematical equations that have an iterative manner. Chaotic processes are not random; they follow rules, but even very simple rules can produce extreme complexity. This complexity can be expressed by a series of equations or visualized and rendered when the element of color is introduced into its interpretation. The mathematics of chaos provides the tools for creating and displaying such phenomenon. It is a way to measure the degree of harshness, unevenness, irregularity of a given object. i.e.: Objects that described Chaos were irregular in shape, ripped and unpredicted[2].

Fractal geometry can be defined as a geometry whose dimension is not an integer, in contrast with the elements of the classic geometry whose dimension is always integers (1 for lines, 2 for areas, 3 for volumes). This concept can be understood better by observing (Fig. 3.1) of the Cantor set and the Koch curve. In both sets there are series of paradoxes that can not be solved using classic geometry.

[1] Ibid.
[2] Ibid.

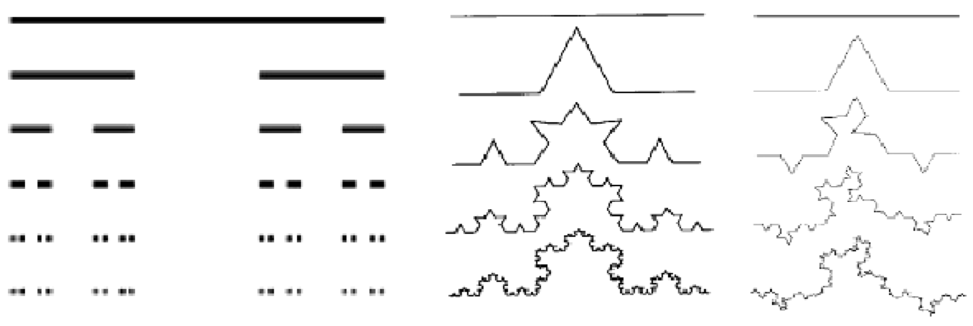


Fig. 3.1: Cantor set (left) and Koch curve (right)[1].

For example the **Cantor set**, each line is substituted for two others, each one of them is 1/3 of the preceding. If the substitution continues indefinitely, what kind of structure will remain: segments or points? The answer is not obvious for the classic geometry, now that it has a non entire dimension, a hybrid between a point and a line.

Koch curve presents another paradox on its genesis: each time that a segment line is substituted by another 4, its length grows up 1/3. So, given that this object is created after infinite substitutions, its length is infinite.

3-1-2 Classifications

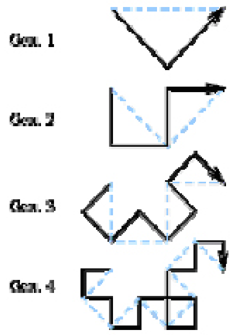
Various types using different algorithms have been discovered to produce fractals like: Ant Automaton, Barnsley Mandelbrot/Julia Set, Bifurcation, Martin Attractors, Circle, Peterson Variations, Formula, Diffusion Type, Dynamic System, Pick over Popcorn, Frothy Basins, Gingerbreadman, Halley, Hyper Complex, Newton, Icon and Icon3d ,Julia Sets ,Inverse Julia's, L-Systems (2d, 3d).

They can be classified into three main categories depending on the technique they are generated from and the mathematical process that are used to calculate them[2].

[1] Bovill, C.: *Fractal Geometry in Architecture and Design*. Birkhauser, Boston 1996.
[2] Ibrahim, M. and Krawczyk, R.J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA, February 2001.

3-1-2-1 Vector Fractals

They are generated from the collection of vector substitution, Like Koch snowflake, cantor set, Barnesley's Fern, and the Dragon curve (Fig. 3.2, 3.3, 3.4)[1]. Fractal generates from any set of vectors or any defined curve.



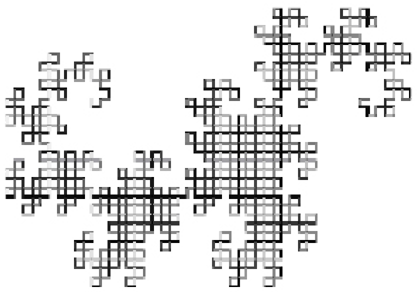


Fig. 3.2: First four generations of Dragon Curve. Fig. 3.3: Dragon Curve (vector fractal).

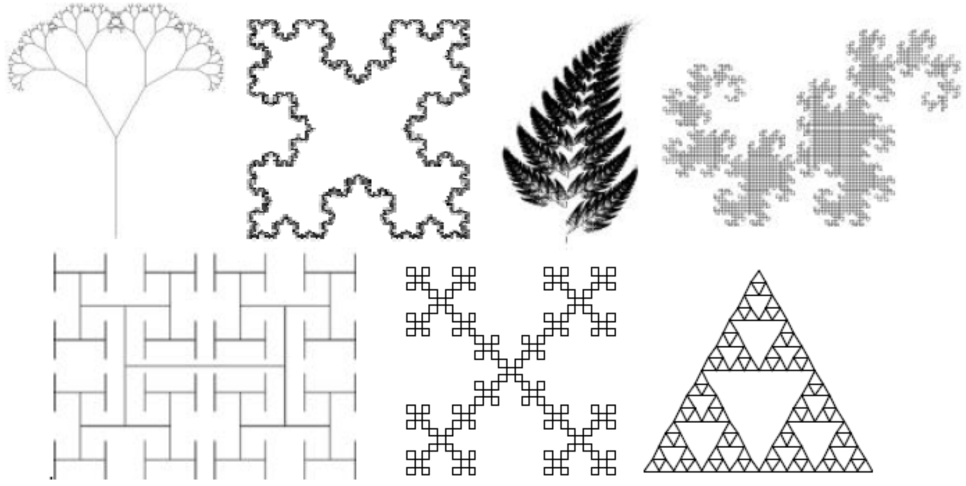


Fig. 3.4: Form generated using vector-base fractal; Tree fractal, Cesàro fractal, Barnsley's Fern, Dragon Curve, H-fractal, Sierpinski square and triangle[2].

[1] Ibid.
[2] Some fractals that exhibit a vector quality can also be generated by point plotting methods. P.93.

3-1-2-2 Point Fractals

They are groups of points in a complex plane like the Mandelbrot set and the Julia set. They are Two-dimensional (Fig. 3.5, 3.6), three-dimensional (Fig. 3.7) or multiple dimensional.

They represent a single case of the IFS that is using the complex numbers or the hyper complex numbers in a Cartesian plane to plot the fractals. The Mandelbrot set and Julia set are examples[1].

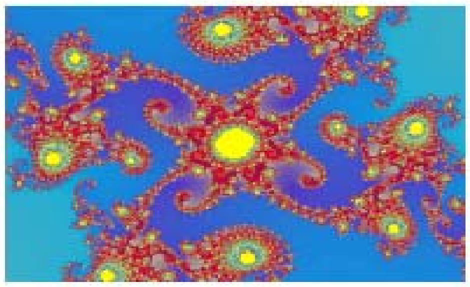


Fig. 3.5: 2D Julia set.

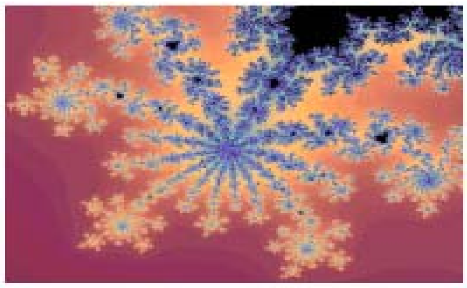


Fig. 3. 6: 2D Mandelbrot set.

Transforming or extruding existing 2d fractals like Mandelbrot or Julia sets along z axis, using series of parameters in a programming code enables the designer to easily manipulate and generate forms in 3 dimensions, they are called "Quaternions"[2].

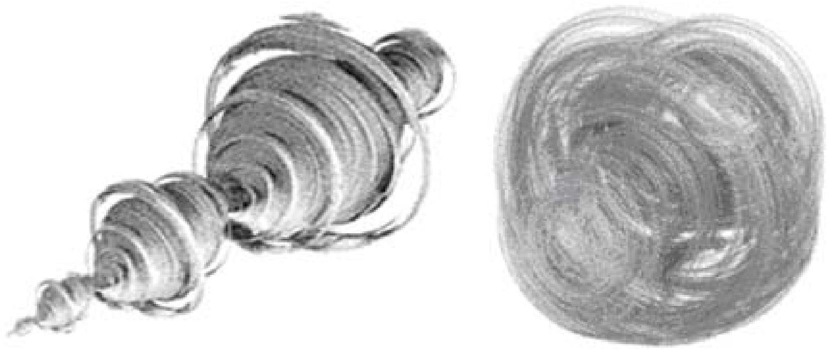


Fig. 3.7: 3D Quaternion Julia set generated by Pov-Ray.

[1] Ibrahim, M. and Krawczyk, R.J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA. February 2001, op.cit.
[2] Mohammadi, G.: *Geometric Shape Generator*. M.Sc. thesis, University of Washington, 2004, op.cit.

(Fig. 3.8) shows some of the most well-known complex fractal types.

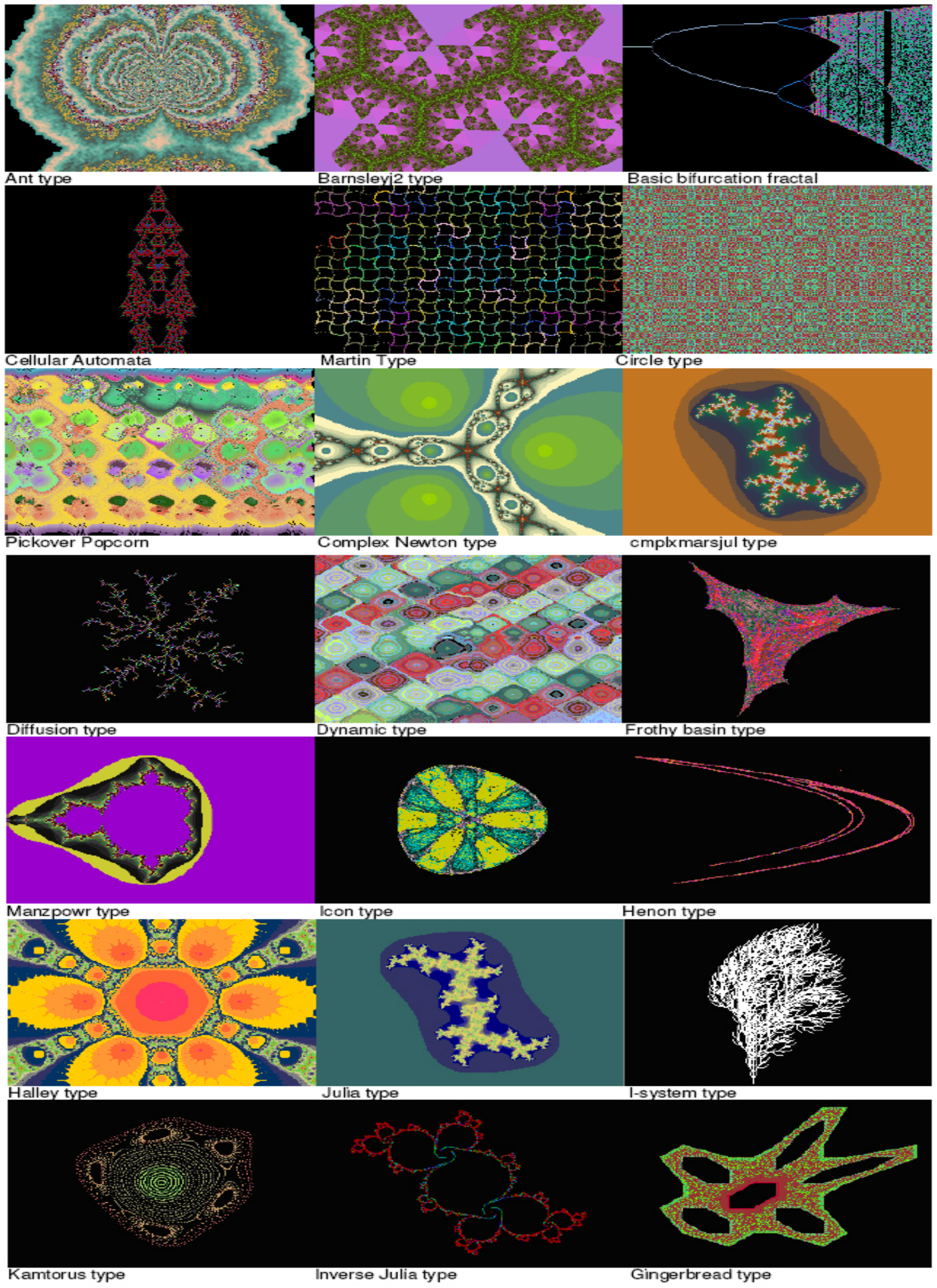


Fig. 3.8: Shapes generated with various complex 2D fractal types[1].

[1] Ibid.

3-1-2-3 Orbit Fractals (Strange Attractors)

Plotting an orbit path in two and three-dimension generates this fractal space. Examples include the Bifurcation orbit, Lorenz Attractors, Rossler Attractors, Henon Attractors, Pickover Attractors, Gingerbreadman, and Martin Attractors. They can be divided into two types; two dimensional attractors and three Dimensional attractors[1].

ii- 2D Attractors

Clifford Pickover[2] developed a two-dimensional strange attractors based on a simple equation consisting of sine functions. What are intriguing about these attractors were the variations possible by the execution of a simple iterative mapping with minor changes in parameters. (Fig. 3.9) displays two such attractors based on equation 1.

$$x_{t+1} = \sin(by_t) + c\,\sin(bx_t) \qquad \text{(Equation 1)}$$
$$y_{t+1} = \sin(ax_t) + d\,\sin(ay_t)$$

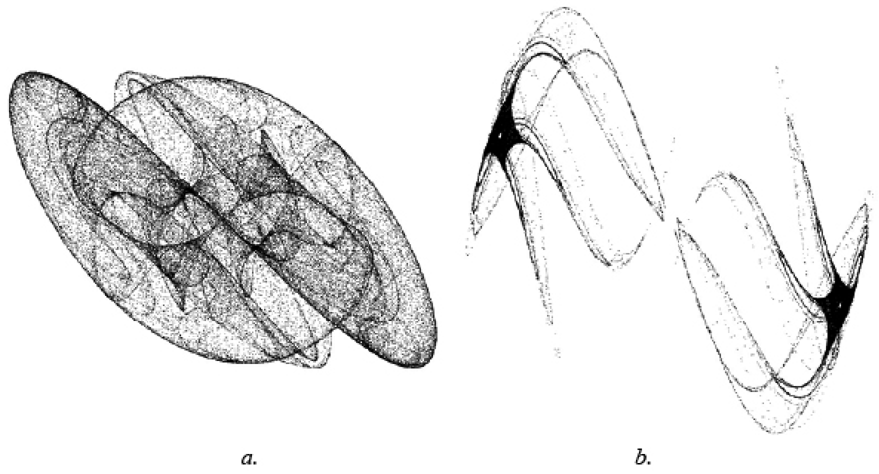


Fig. 3.9: Pickover's 2D strange attractors[3].

[1] Ibrahim, M. and Krawczyk, R.J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA. February 2001, op.cit.
[2] Pickover, C.: *Chaos in Wonderland*. St.Martin's Press, New York, 1994.
[3] Krawczyk, R.J.: *Dimension of Time in Strange Attractors*. College of Architecture, Illinois Institute of Technology, USA. ISAMA, Bridges Conference, 2003.

Paul Bourke[1] developed another equation. This equation is a variation of the ones that Pickover developed. In reproducing these images, a time limit of 100,000 computations was used. This single attractor displays some interesting visual aspects depending on the values selected (Fig. 3.10).

$$x_{t+1} = \sin(ay_t) - \cos(bx_t) \qquad \text{(Equation 2)}$$
$$y_{t+1} = \sin(cx_t) - \cos(dy_t)$$

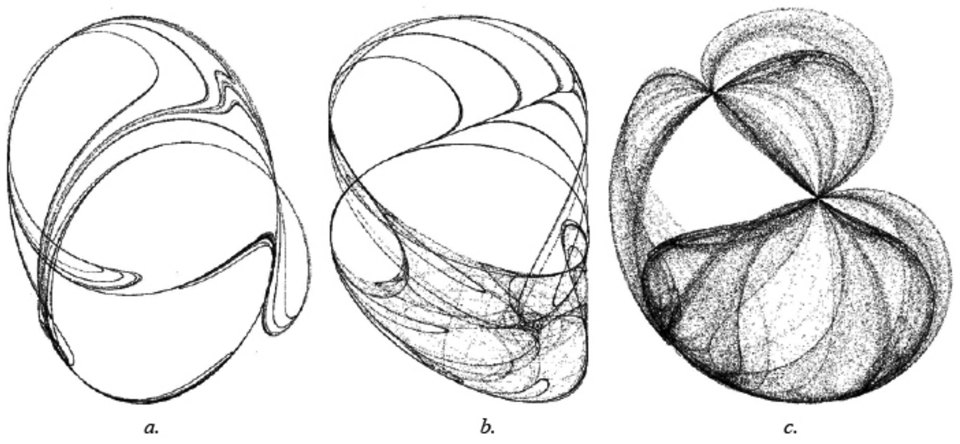


Fig. 3.10: Bourke's 2D strange attractor[2].

In the first, all the points lie on curves with little or no deviation. In the second, a distinct curve is also traced but you can begin to see lighter trails forming. In the third, the distribution of points is much greater and because of the density of points in certain areas, surfaces begin to appear (virtually). The curves which are greater congregation of points begin to represent the edges of these surfaces.

A third dimension appearance is only implied by these visual cues since the equations are a two-dimensional mapping. In this case density of points is controlled by time or time seen to begin to give two-dimensional attractors a three dimensional visual appearance. The distance between the points controls apparent shading.

[1] Bourke, P.: *The Pattern Book: Fractals, Art and Nature*. Edited by Pickover C., World Scientific Publishing, Singapore, 1995.
[2] Krawczyk, R.J.: *Dimension of Time in Strange Attractors*. College of Architecture, Illinois Institute of Technology, USA. ISAMA, Bridges Conference, 2003. op.cit.

When time is increased, the density of the points further draws out either hidden areas within regions of the attractor or it strengthens what appear to be edges of three-dimensional surfaces. (Fig. 3.11) displays the Bourke's third attractor using 200,000 points. Increase of time will generate finer detail at a given image resolution[1].

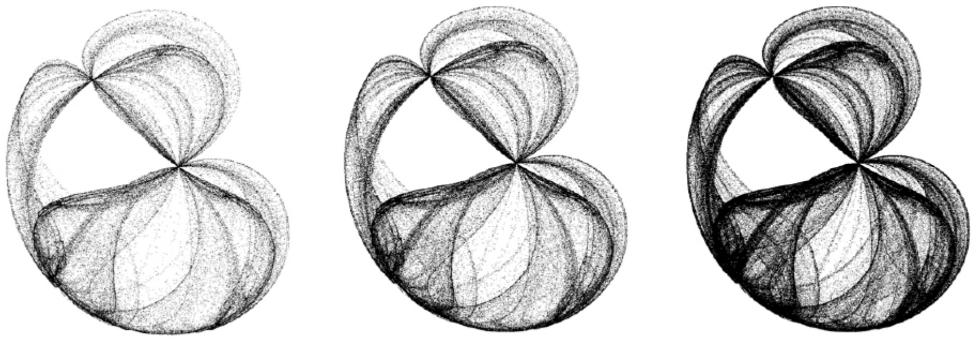


Fig. 3.11: Comparison of density.

(Fig. 3.12 a) displays another Bourke attractor based on equation 2. This particular attractor is one that generates a large number of repeating values so that distinct curves are created with a small number of points appearing between the curves. Computing more points, adding time, will not introduce any inbetween points. Devaney[2] discussed a random iteration algorithm for iterated function systems that seemed appropriate for expanding the generated curves.

This method is based on the random selection from a range of specific values. For this attractor this basic concept was used in two variations. The first, (Fig. 3.12 b) used a percent offset to compute related attractors. From these computes values any one was selected. The result is a soft haze surrounding each of the curves in the attractor. The second (Fig. 3.12 c) used the same percent offset, but the possible random values were selected from a fixed interval. The haze surrounding the curves

[1] Ibid.
[2] Devaney, R.*: Chaos and Fractals, The Mathematics behind the Computer Graphics*. American Mathematical Society, Providence, p.141. 1989.

was greater in area. In both cases, the regions between the curves begin to be filled and create possible surfaces.

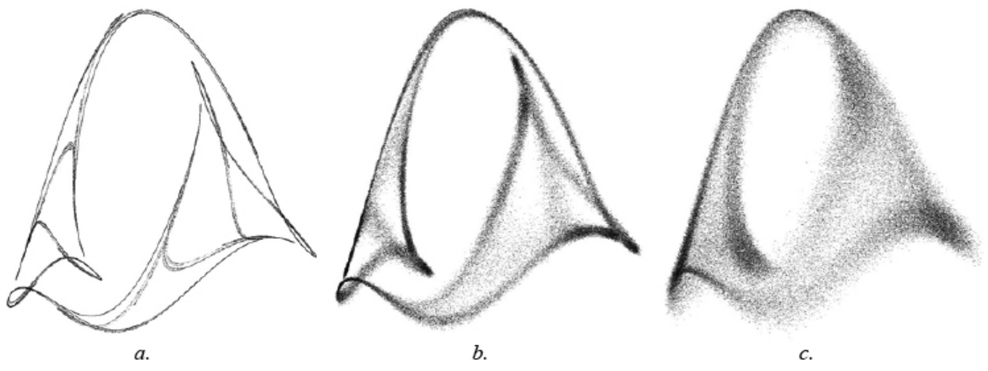


Fig. 3.12: Random selection of values for an attractor.

While selecting values for a variety of attractors, it was noticed that related images were created when a single parameter was modified. For example, the attractor based on equation 3 (Fig. 3.13) can generate a series having an unraveling motion when only the first parameter is changed. A series such as this can be viewed as a static images, frozen time, or animated to give an attractor the added dimension of motion. Other attractors were found that developed an unfolding motion. The challenge is to determine in a specific attractor which parameter or parameters can be changed and their range to generate this effect. It does offer a unique display of related family attractors.

$$x_{t+1} = |\sin^3(ay_t)| + \cos(bx_t) \qquad \text{(Equation 3)}$$
$$y_{t+1} = |\sin^2(cx_t)| - \cos^2(dy_t)$$

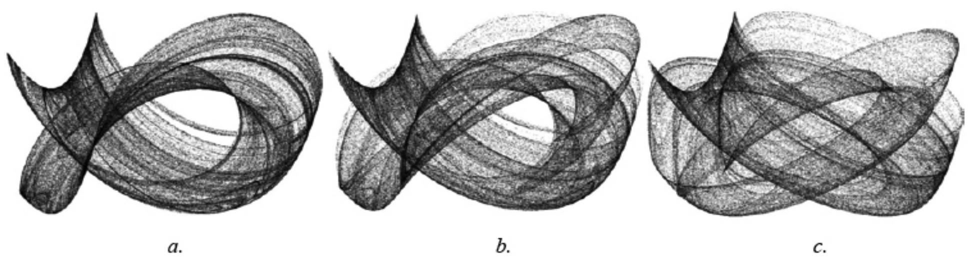


Fig. 3.13: Unraveling an attractor[1].

[1] Krawczyk, R.J.: *Dimension of Time in Strange Attractors*. College of Architecture, Illinois Institute of Technology, USA. ISAMA, Bridges Conference, 2003. op.cit.

(Fig. 3.14) displays a sample from the current series of rendered attractors based on this concept. Many of these could have been inspired from natural forces, such as, wind, smoke water, earthen formations and other folding, bending, twisting, draping and crumpling of identifiable materials or organisms. The third dimension is determined by the perception of the viewer coupled with the created intent by selection of attractor and its parameters.

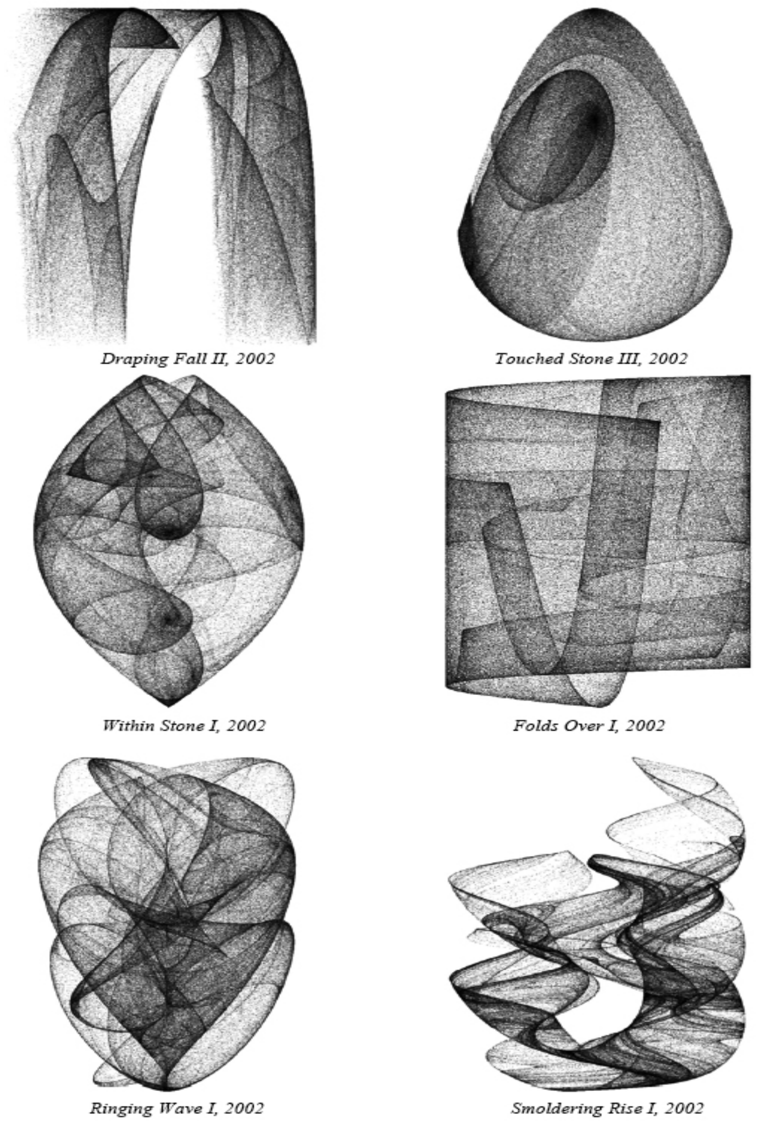


Fig. 3.14: Examples of completed strange attractors[1].

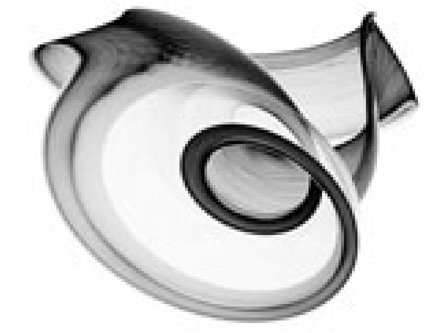
[1] Ibid.

iii- 3D Attractors

Many 3D attractors had been developed, (Fig. 3.15 – 3.27) show a rendered result and description for some of these strange attractors with corresponding mathematical function[1].

- **Chaotic Flow**

In this equation the position of the variables (x, y and z) is itself a parameter, Mi Op. This means that during a Search, not only the parameters are randomized, so is the equation.



22 parameters : **M0, M1... M11, M0 Op., M1 Op.... M10 Op** and **dT**

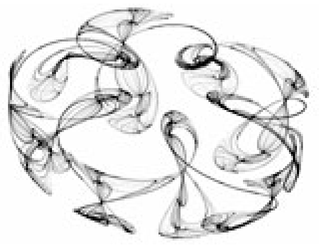
equation : $Op_i \in \{1, x, y, z\}$

$$\dot{v} = v + d\left(\begin{bmatrix} m_0 Op_0 & m_1 Op_1 & m_2 Op_2 \\ m_4 Op_4 & m_5 Op_5 & m_6 Op_6 \\ m_8 Op_8 & m_9 Op_9 & m_{10} Op_{10} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} m_3 \\ m_7 \\ m_{11} \end{bmatrix}\right)$$

Fig. 3.15: Chaotic flow strange attractor.

- **Icon**

This attractor is an example of a two dimensional attractor converted to a three dimensional one: values of z are not used for x and y calculations.



6 parameters : **Degree**, **Alpha**, **Beta**, **Lambda**, **Gamma** and **Omega**

equation :

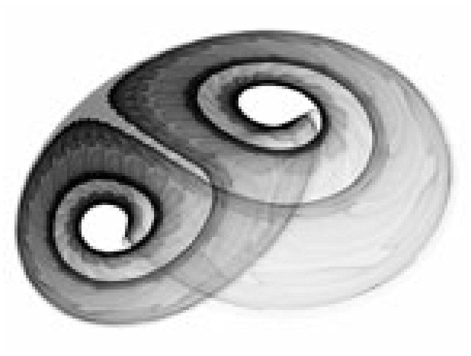
$$r \in \mathbf{C}$$
$$r = (x + iy)^d$$
$$p = \lambda + \alpha\|v\| + \beta(x \operatorname{Re} r - y \operatorname{Im} r)$$
$$\dot{x} = px + \gamma \operatorname{Re} r - \omega y$$
$$\dot{y} = py - \gamma \operatorname{Im} r + \omega x$$
$$\dot{z} = \|v\|$$

Fig. 3.16: Icon strange attractor.

[1] http://www.btinternet.com/~ndesprez/manual/attractors.htm. Accessed 23/10/2007.

- **Lorenz**

Simple equation created by Edward Lorenz to demonstrate the chaotic behavior of dynamic systems Lorenz discovered, while working on weather simulation, one of the fundamental laws of the Chaos Theory: "the sensitive dependence on initial conditions" he himself dubbed "the butterfly effect".



4 parameters : **A**, **B**, **C** and **dT**

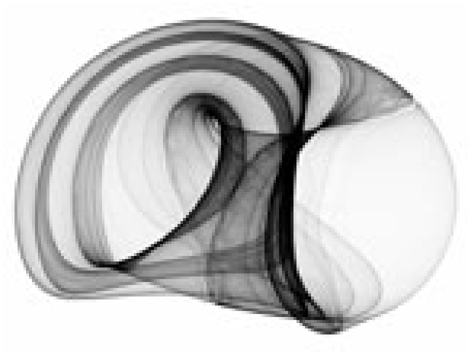
equation :

$$\dot{x} = x + ad(y - x)$$
$$\dot{y} = y + d(bx - y - zx)$$
$$\dot{z} = z + d(xy - cz)$$

Fig. 3.17: Lorenz strange attractor.

- **Lorenz-84**

This equation is a low-dimensional model for long term atmospheric circulation. Rather than a graphical representation of atmospheric currents, the orbit coordinate are the three variables of the model.



5 parameters : **A**, **B**, **F**, **G** and **dT**

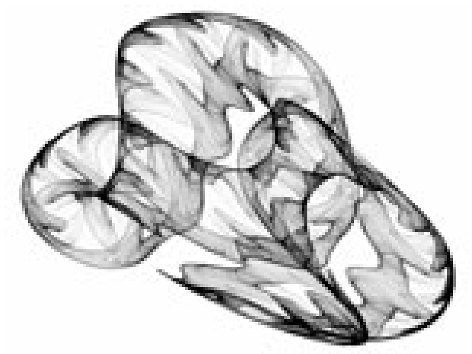
equation :

$$\dot{x} = x + d(-ax - y^2 - z^2 + af)$$
$$\dot{y} = y + d(-y + xy - bxz + g)$$
$$\dot{z} = z + d(-z + bxy + xz)$$

Fig. 3.18: Lorenz-84 strange attractor.

- **Pickover**

Probably the best equation to start experimenting with, since the orbit will never escape the attractor as it is "trapped" in sinusoids. A good set of parameters is {1, 1.8, 0.71, 1.51} from which nice attractors can be found, after modifying each parameter slightly.



4 parameters : **A, B, C and D**

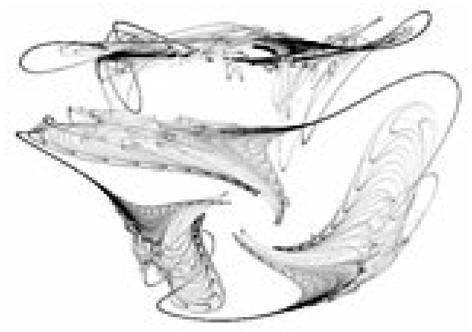
equation :

$$\dot{x} = \sin ay - z\cos bx$$
$$\dot{y} = z\sin cx - \cos dy$$
$$\dot{z} = \sin x$$

Fig. 3.19: Pickover strange attractor.

- **Polynomial, Type A**

The Polynomial type A uses the simplest equation, it is therefore the fastest attractor to render. The equation is a special case of Type B where P1, P3 and P5 = 0.



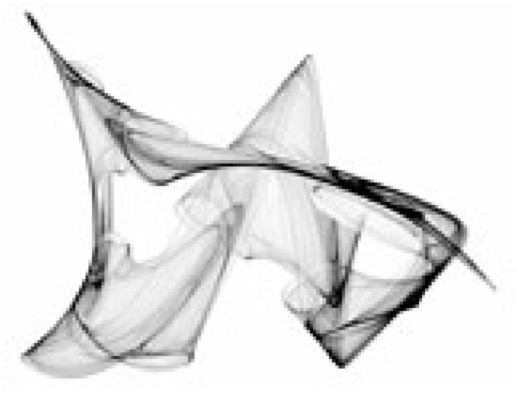
3 parameters : **P0, P1 and P2**

equation :

$$\dot{x} = p_0 + y - zy$$
$$\dot{y} = p_1 + z - xz$$
$$\dot{z} = p_2 + x - yx$$

Fig. 3.20: Polynomial, type a strange attractor.

- **Polynomial, Type B**



A slightly more complex equation for a challenging specimen. This attractor equation is rather feeble and will see the orbit escapes more than often. When not escaping, it won't produce a great variety of shapes.

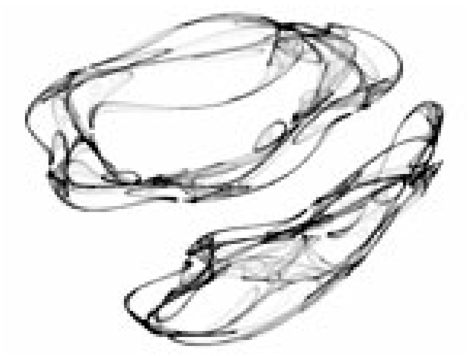
6 parameters : **P0, P1... P5**

equation :

$$\dot{x} = p_0 + y - z(p_1 + y)$$
$$\dot{y} = p_2 + z - x(p_3 + z)$$
$$\dot{z} = p_4 + x - y(p_5 + x)$$

Fig. 3.21: Polynomial, type B strange attractor.

- **Polynomial, Type C**



Next step up the Polynomial evolution ladder, it has three times more parameters than its predecessor. This equation is a good compromise between speed and complexity.

18 parameters : **P0, P1... P17**

equation :

$$\dot{x} = p_0 + x(p_1 + p_2 x + p_3 y) + y(p_4 + p_5 y)$$
$$\dot{y} = p_6 + y(p_7 + p_8 y + p_9 z) + z(p_{10} + p_{11} z)$$
$$\dot{z} = p_{12} + z(p_{13} + p_{14} z + p_{15} x) + x(p_{16} + p_{17} x)$$

Fig. 3.22: Polynomial type c strange attractor.

- **Polynomial Function (Abs, Power and Sin)**

Polynomial Function is a group of three equations which make use of algebraic or trigonometric functions rather than the normal 2nd order structure. They were adapted to 3 dimensions from Julien Sprott's book on Strange Attractors. Abs (Fig.3.23) will yield very typical angular forms, Power (fig. 2) will add some flexibility to Abs straight lines, while Sin (fig. 3) will produce wavy attractors.

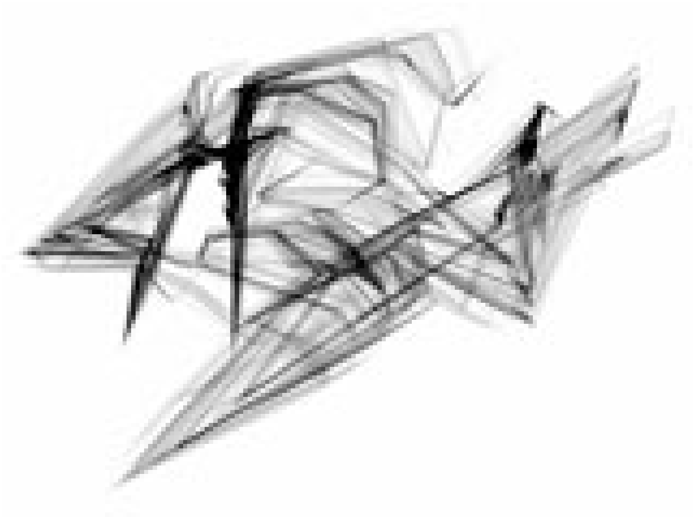
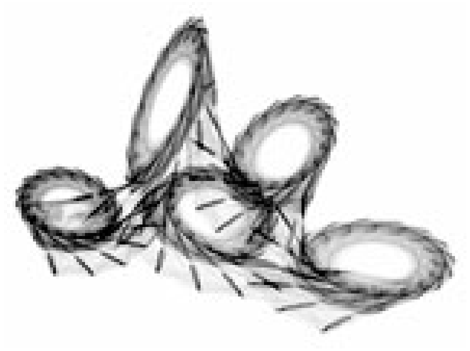


Fig. 3.23: Polynomial function –Abs.



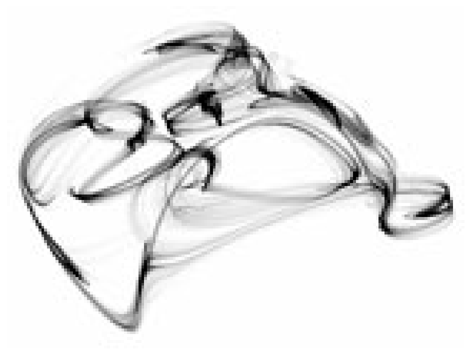


Fig. 3.24: Polynomial function-Power.

Fig. 3.25: Polynomial function-Sin.

21 to
39 parameters : **P0, P1... P38**

equations :

Abs:

$$\dot{x} = P_0 + P_1x + P_2y + P_3z + P_4|x| + P_5|y| + P_6|z|$$
$$\dot{y} = P_7 + P_8x + P_9y + P_{10}z + P_{11}|x| + P_{12}|y| + P_{13}|z|$$
$$\dot{z} = P_{14} + P_{15}x + P_{16}y + P_{17}z + P_{18}|x| + P_{19}|y| + P_{20}|z|$$

Power:

$$\dot{x} = P_0 + P_1x + P_2y + P_3z + P_4|x| + P_5|y| + P_6|z|^{P7}$$
$$\dot{y} = P_8 + P_9x + P_{10}y + P_{11}z + P_{12}|x| + P_{13}|y| + P_{14}|z|^{P15}$$
$$\dot{z} = P_{16} + P_{17}x + P_{18}y + P_{19}z + P_{20}|x| + P_{21}|y| + P_{22}|z|^{P23}$$

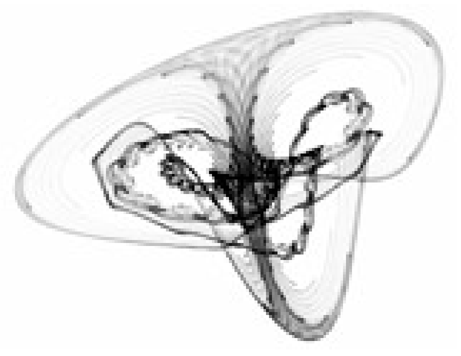
Sin:

$$\dot{x} = P_0 + P_1x + P_2y + P_3z + P_4\sin(P_5P_6x) + P_7\sin(P_8P_9y) + P_{10}\sin(P_{11}P_{12}z)$$
$$\dot{y} = P_{13} + P_{14}x + P_{15}y + P_{16}z + P_{17}\sin(P_{18}P_{19}x) + P_{20}\sin(P_{21}P_{22}y) + P_{23}\sin(P_{24}P_{25}z)$$
$$\dot{z} = P_{26} + P_{27}x + P_{28}y + P_{29}z + P_{30}\sin(P_{31}P_{32}x) + P_{33}\sin(P_{34}P_{35}y) + P_{36}\sin(P_{37}P_{38}z)$$

Fig. 3.26: The three polynomial function equations.

- **Polynomial, Sprott**

It is the most complex equation found. Given the number of parameters, it allows 20^{30} combinations, which means it would have virtually one chance out of 1,073,741,824,000,000, 000,000,000,000,000,000,000,000 to reproduce a strange attractor.



up to
168 parameters : **P0, P1... P167**

equation : (equation for the 2nd order)

$$\dot{x} = P_0 + P_1x + P_2x^2 + P_3xy + P_4xz + P_5y + P_6y^2 + P_7yz + P_8z + P_9z^2$$
$$\dot{y} = P_{10} + P_{11}x + P_{12}x^2 + P_{13}xy + P_{14}xz + P_{15}y + P_{16}y^2 + P_{17}yz + P_{18}z + P_{19}z^2$$
$$\dot{z} = P_{20} + P_{21}x + P_{22}x^2 + P_{23}xy + P_{24}xz + P_{25}y + P_{26}y^2 + P_{27}yz + P_{28}z + P_{29}z^2$$

Fig. 3.27: Polynomial sprott strange attractor.

3-1-3 Architectural Potentials

3-1-3-1 Generating 2D Forms

i- Using Vector-Base Fractal[1]

The vector-base fractals can be used to generate a wide variety of 2D shapes and patterns by applying the fractal process on a selected initiator and a generator (Fig. 3.28).

A vector-base fractal is composed of two parts: the initiator and the generator. In the Kock curve for example, the generator is a line that is divided into three equal segments, and the middle segment forms an equilateral triangle. By replacing every line of the initiator with the full generator, we get the first iteration of the snowflake.

By iterating this operation again and again, replacing every line of the new initiator with the full generator, it ends with a figure that approximates a snowflake. The iteration process should continue to infinity to generate a real Koch Snowflake fractal, but the function is iterated for some finite number of times. (Fig. 3.29) displays the Koch Snowflake with 3 iterations.

If the generator is changed, inverted, an entirely different form can be developed, like the Koch Antisnowflake shown in Figure 10.

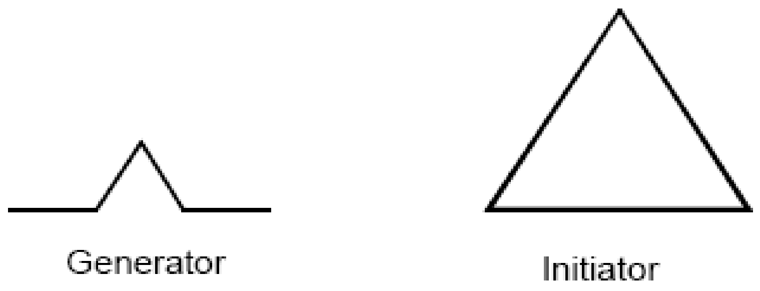


Fig. 3.28: Generator and initiator.

[1] Ibrahim, M. and Krawczyk, R.J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA, February 2001, op.cit.

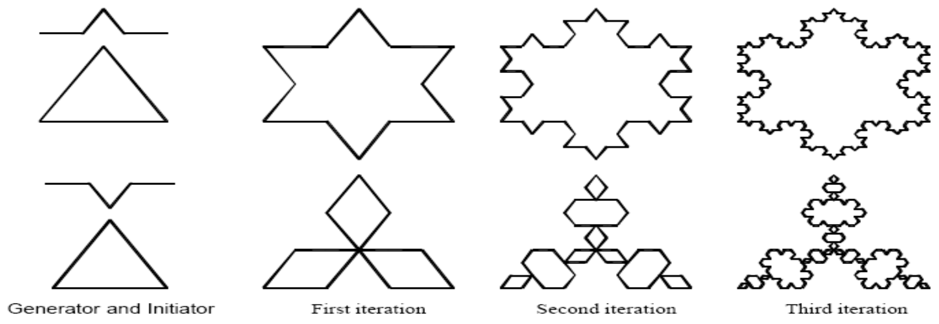


Fig. 3.29: The Koch Snowflake and the Antisnowflake.

Some of the IFS fractals (vector fractals) are: Cantor Set, Barnsley's Fern, Koch Antisnowflake, Koch Snowflake, Box Fractal, Cantor Square Fractal, Cesàro Fractal, Dragon Curve, Gosper Island Fractal, H-Fractal, Sierpinski Curve, Minkowski Sausage.

All of these fractals are based on simple geometric shapes: lines, squares, or triangles. Shakiban and Berstedt[1] discussed a new generating procedure based on vector calculus and modular arithmetic to generate the Koch Snowflake. The procedure was then applied to create more generalized snowflakes rather than the triangular classical snowflakes. They also suggested the use of n-sided polygons, such as pentagons as initiators.

- Another method to modify the geometric shapes produced by a fractal comes from the random selection of the direction and displacement of the initiator.

- Direction and Proportion Variations[2]

Normally, the length of the generator is equal to a segment of the initiator and the direction of the line segments in the generator and initiator are the same as seen in the Koch Snowflake in (Fig.3.29).

[1] Shakiban, C. and Berstedt, J.E.: *Generalized Koch Snowflakes*. In Bridges: Mathematical Connections in Art, Music and Science, 1998.
[2] Ibrahim, M. and Krawczyk, R.J.: *Exploring the Effect of Direction on Vector-Based Fractals.* College of Architecture, Illinois Institute of Technology, USA, Bridges 2002 conference, Towson, MD, July, 2002.

One possible variation is to modify the direction of the entire generator or initiator or individual parts of each and the orientation of the generator as it is placed on the initiator.

(Fig. 3.30) displays the Koch Snowflake with the generator in one direction and normal and reversed direction for the initiator. When the initiator is reversed the Antisnowflake appears. (Fig. 3.31) reverses the direction of the placement of the generator. The fractals developed starting at the second iteration are undocumented versions of the Koch Snowflake. Additional variations were found by modifying the direction of individual line segments within the generator or the initiator.

In addition to direction, the proportions of the line segments in the generator could be modified in relationship to the initiator. To demonstrate this concept, a square initiator and the normal Koch Snowflake generator are used. (Fig. 3.32) shows fractals that are based on generators that range from 25% to 75% of the size of the initiator.

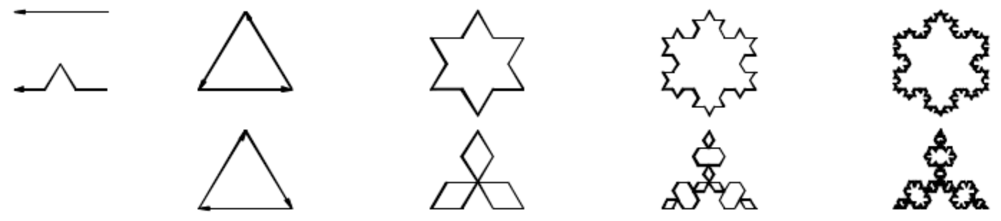


Fig. 3.30: The direction effect on the generated fractal applied on the simple Koch Snowflake.

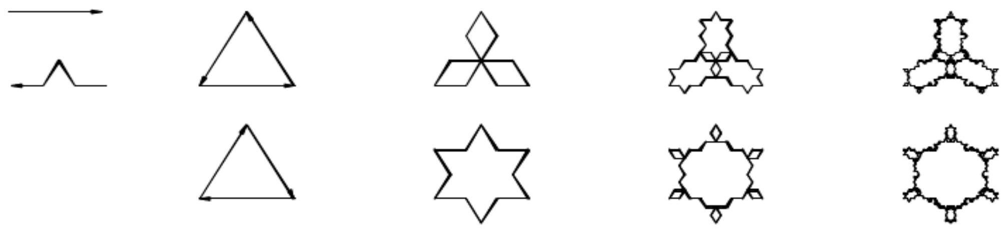


Fig. 3.31: The effect of the direction of the placement of the generator on the generated fractal applied on the simple Koch Snowflake.

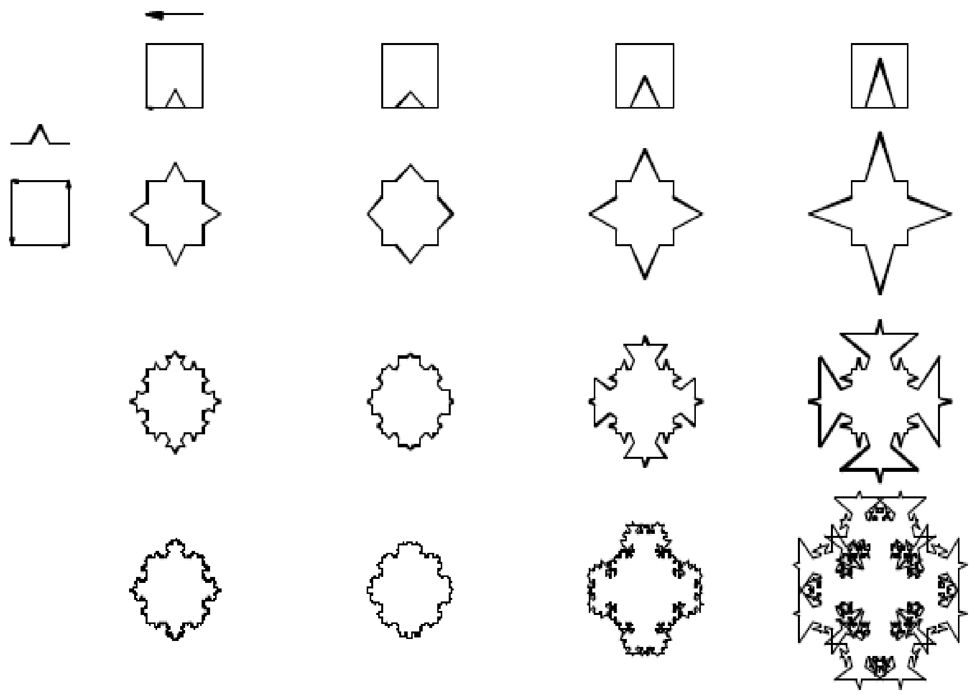


Fig. 3.32: The proportion effect on the generated fractal applied on the simple square.

The classical fractals have no specific meaning associated to their shapes; they are simply forms that generate interesting fractals. Selecting the generator can be based on architectural organizational schemes used as major axes of a site planning or a building, or using the patterns discussed by Francis D.K. Ching in organizing spaces[1].

The fractal process, if left unrestrained, will go on for ever. It will create an interesting shape but will never produce a building. A building typically has to respond to a multiplicity of process, superimposed or interwoven.

[1] Ibrahim, M. and Krawczyk, R.J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA, February 2001, op.cit.

Therefore, the fractal process needs to be guided, to be constrained and to be filtered. Then has to be mutated by the utilitarian requirements of the functionalities of a building.

The 2D fractals first resulted might be used as site boundaries or circulation axes, or any other architectural organizational element. (Fig. 3.33) shows another base and generator and the first twenty states generated by twenty applications of the generator.

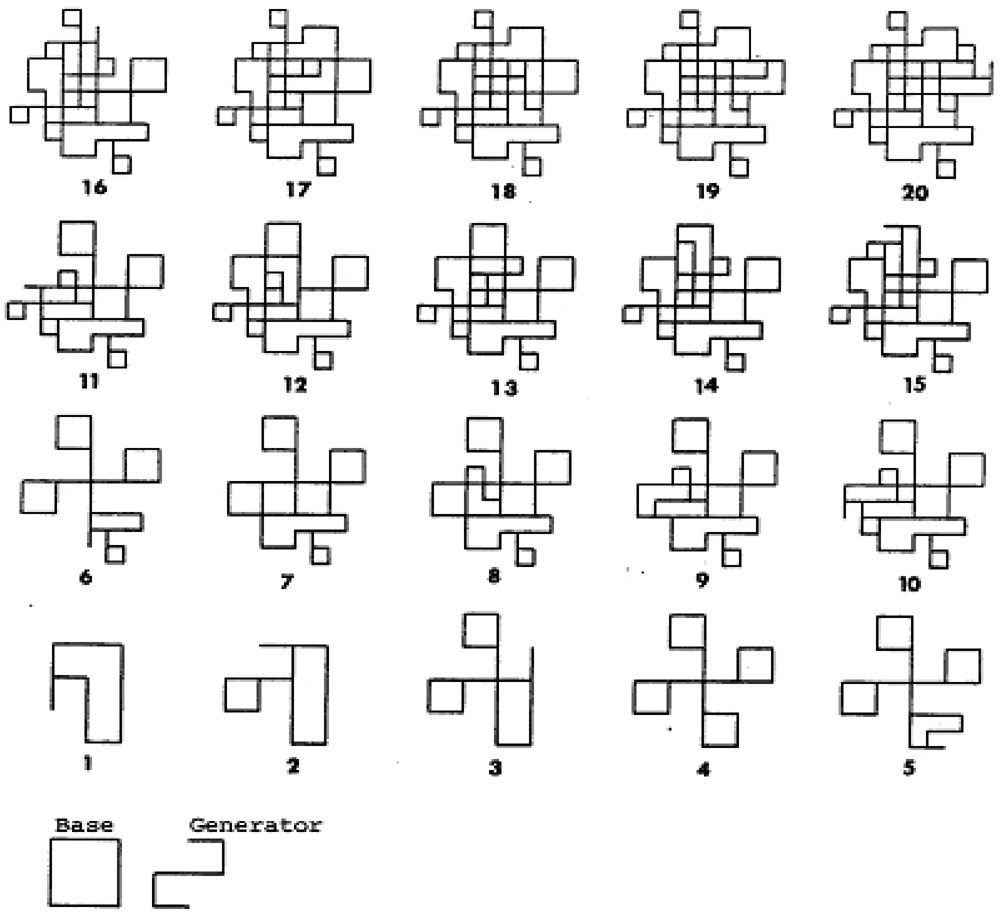


Fig. 3.33: A fractal process after 20 iteration[1].

[1] Yessios, C.I.: *A Fractal Studio*. In ACADIA '87 Workshop Proceedings, 1987.

This concept is further illustrated in (Fig. 3.34). The initial shape (base) can itself be the product of a fractal or some other generative process.

Applying generators I and II individually results to distinctly different shapes. The two generated can then be mixed uniformly to produce the fractal shape in (Fig. 3.34 e). Or the mixture of the generation can be controlled and regulated to produce a shape as the one shown in figure (Fig. 3.34 f).

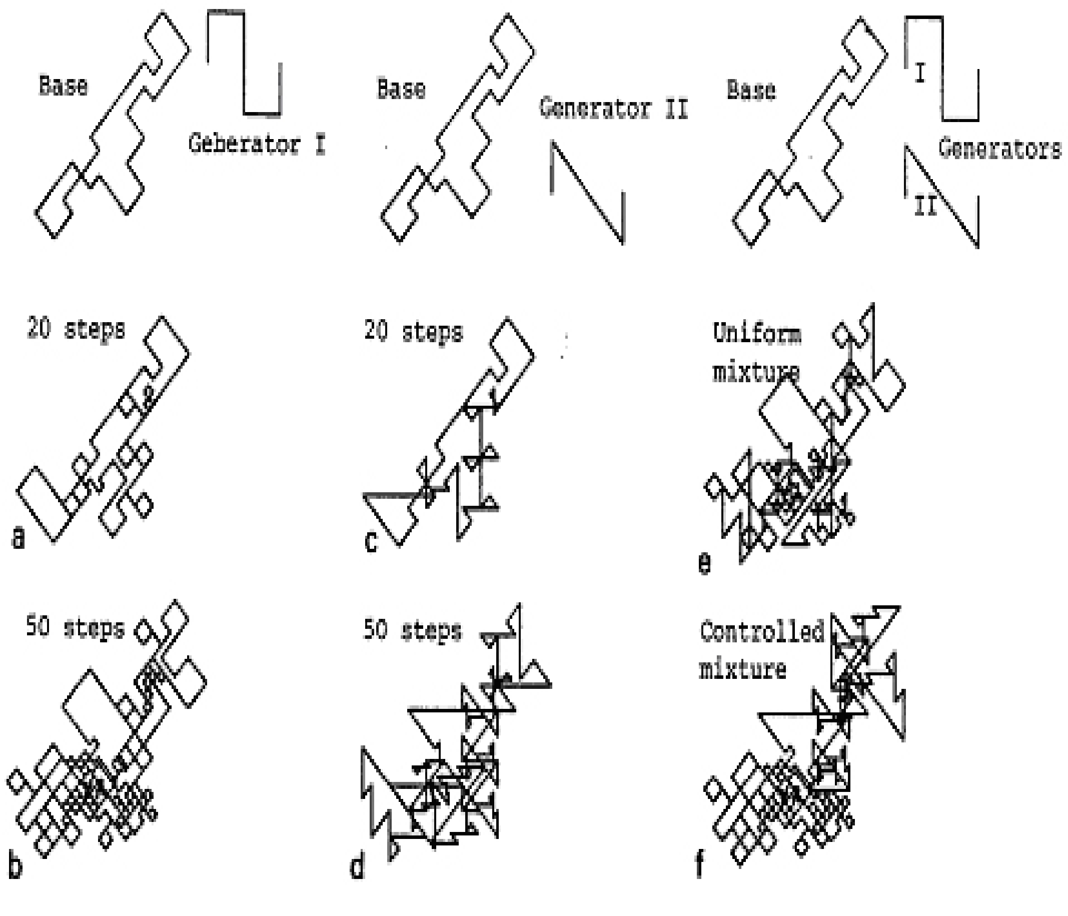


Fig. 3.34: Applying two generators, one at a time & mixed[1].

[1] Ibid.

ii- Using Point Fractal

- Environment Design and Landscape Generation[1]

Point fractals generate an extensive range of shapes. These shapes are especially interesting in the simulation of forests, galaxies, leaves, flowers, rocks, mountains, torrents of water, carpets, bricks and much more. The environment design is one of the most important contributions of fractal geometry to CAAD (Fig. 3.35).

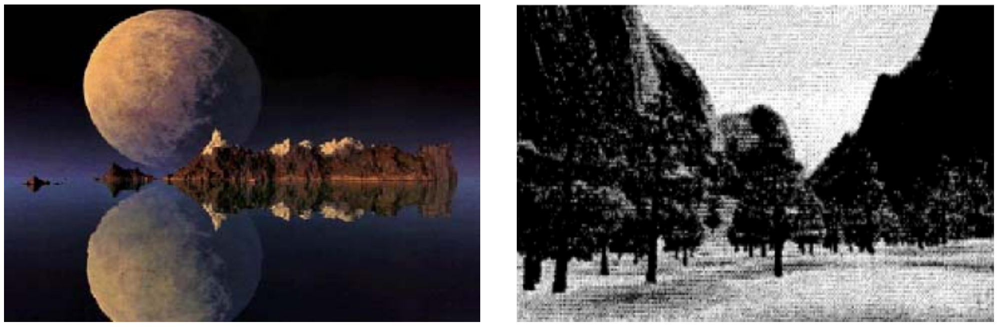


Fig. 3.35: Environment and landscape fractals.

- Texture Generation[2]

The design of fractal texture makes possible the simulation of wood, water, minerals and a long list of materials very useful in photorealistic modeling (Fig. 3.36).

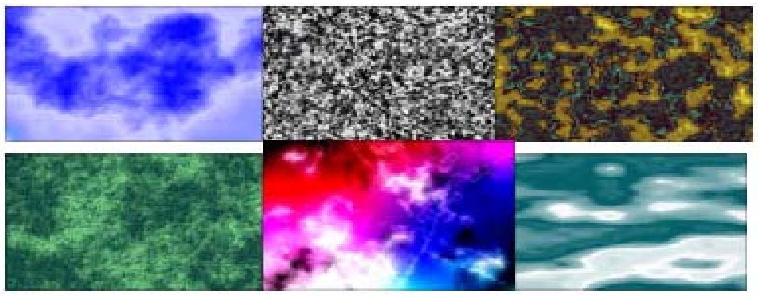


Fig. 3.36: Cloud texture, stucco texture, microscopic texture, granite texture, vegetation texture and water texture.

[1] Garcia, F., Fernandez, A. and Barrallo, J.: *Discovering Fractal Geometry in CAAD*. eCAAD 1994, Proceedings (conversion 2000).
[2] Ibid.

- Patterns Generation

The fractal patterns generated can be used as patterns for architectural layouts or landscape or meshes or terrain or patterns that can be used in architectural building details (Fig. 3.37).

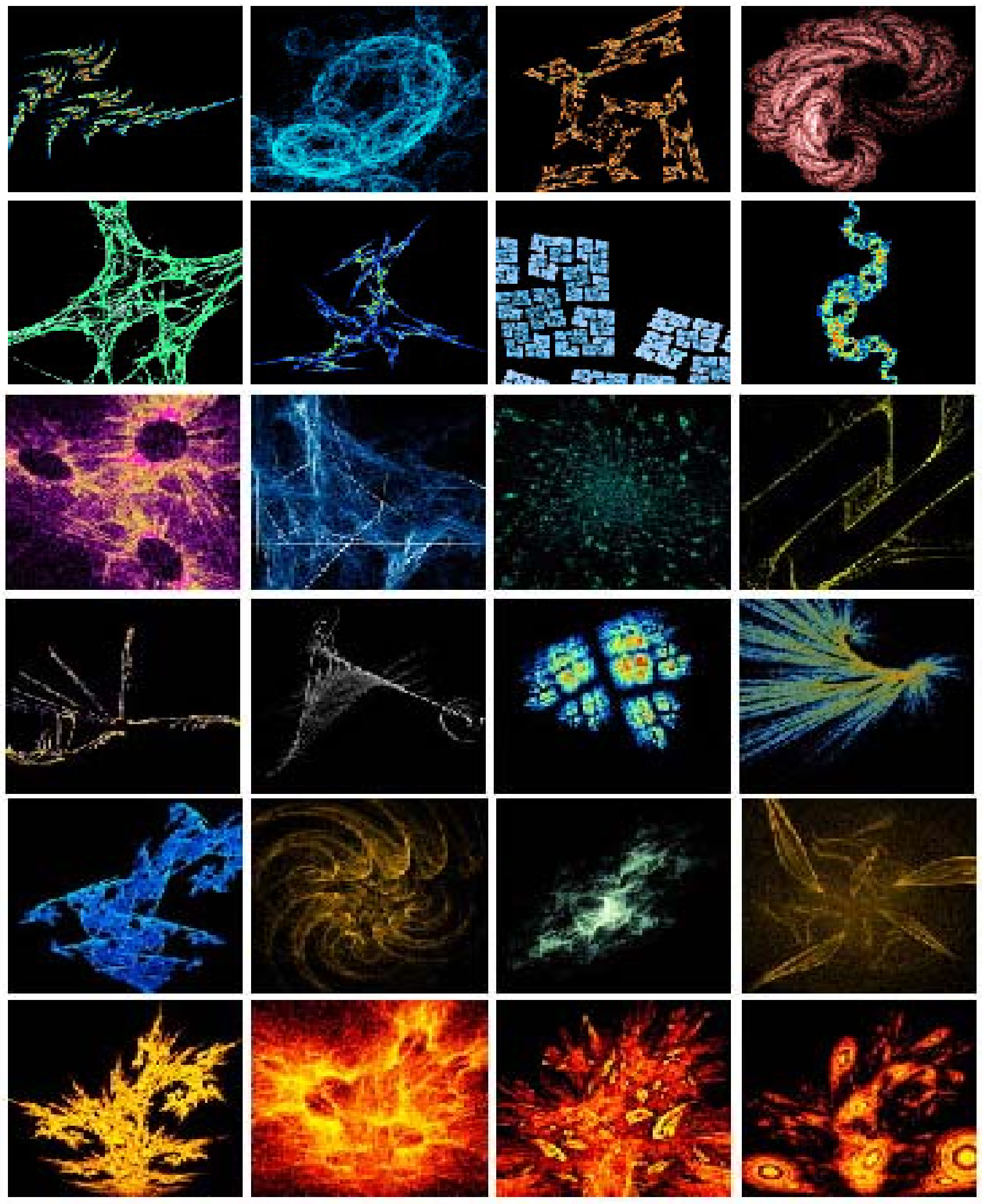


Fig. 3.37: Gallery generated using point fractals[1].

[1] Chapuis, J. and Lutton, E.: *ArtiE-fract: Interactive Evolution of Fractals*. International journal on artificial intelligence tools, 15 December, 2005.

3-1-3-2 Generating 3D Forms

i. Assigning Height to a 2D Vector-Base Fractal

The example below shows how a 2D vector-base fractal can be transformed into 3D vector-base fractal. Yessios's[1] implemented a fractal generation that was highly interactive and allowed a fractal to be developed one iteration at a time or at multiple increments. At the same time, generators could be changed, replaced, deleted, or inserted, at any iteration. The generation process could go forward and backward allowing the designer to return to an earlier state.

In (Fig. 3.38 a) the base and the generator are one and the same shape. The fractal generated after 30 steps (Fig. 3.38 b). It has been filtered and transformed into a structured drawing (floor plan) which next becomes the base for generating a 3D building model. An interior model is shown in (Fig. 3.38 c) and two views of the exterior model are shown in (Fig. 3.38 d) and (Fig. 3.38 e). the heights were determined using Fibonacci sequence (0,1,1,2,3,5,8,13,21,…..).

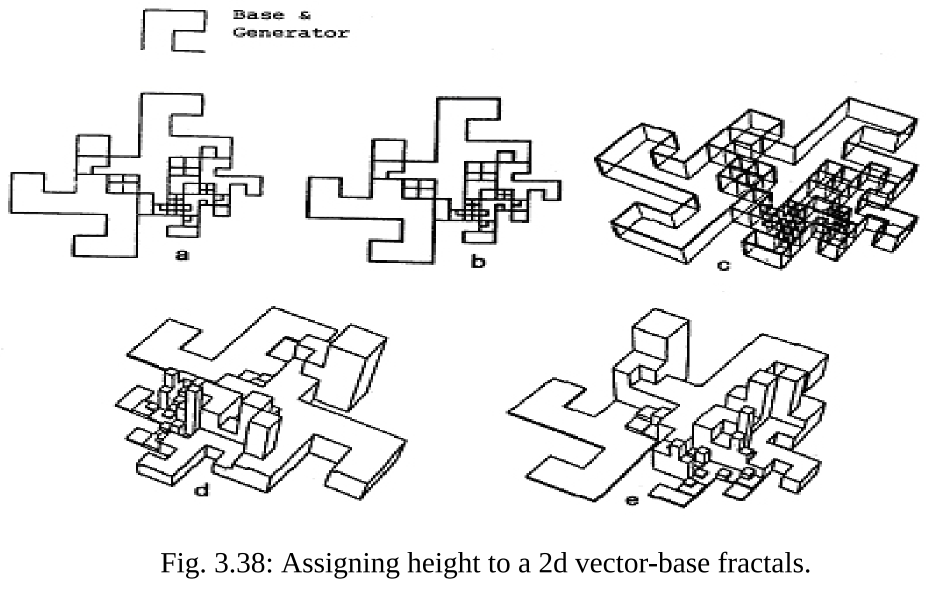


Fig. 3.38: Assigning height to a 2d vector-base fractals.

[1] Yessios, C.I.: *A Fractal Studio*. In ACADIA '87 Workshop Proceedings.1987, op.cit.

ii. Applying a 3D Generator to a 3D axiom

Applying rules to the single or complex 3d axiom results in generating architectural spaces. (Fig. 3.39, 3.40) exemplifies a 3 dimensional structure of the (carpet L-system) fractal. The unit cube is divided to slice the width of the cube by 1/3. The central pieces of the resulting cubes located at the center of the faces are removed from original cube. This process is repeated perpetually with the resulting cubes[1].

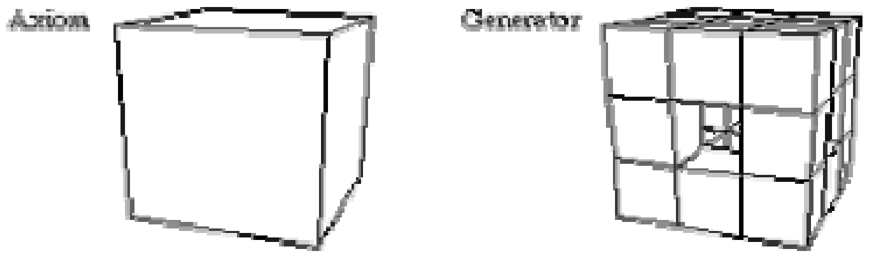


Fig. 3.39: An axiom and a generator.

An example of a cube generated using carpet L-system fractal by applying rules to a simple 3D axiom, results in generating an architectural space (Fig. 3.40).

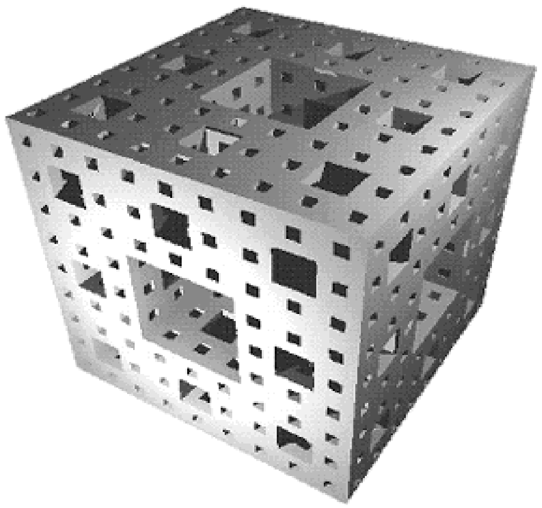


Fig. 3.40: Generated 3 dimensional space using carpet L-system fractal.

[1] Mohammadi, G.: *Geometric Shape Generator*. M.Sc. thesis proposal, University of Washington, 2004..

iii. Metaphorization of Orbit Fractals

Metaphorization of abstract form had been an essential element of creativity. Thus, strange attractors can empower creativeness and trigger inventiveness. (Fig. 3.41) shows a gallery for some rendered attractors that can hold architectural potentials after being mutated by the utilitarian requirements and the functionalities of a building. These attractors resemble forms created by Greg Lynn's Blob architecture.

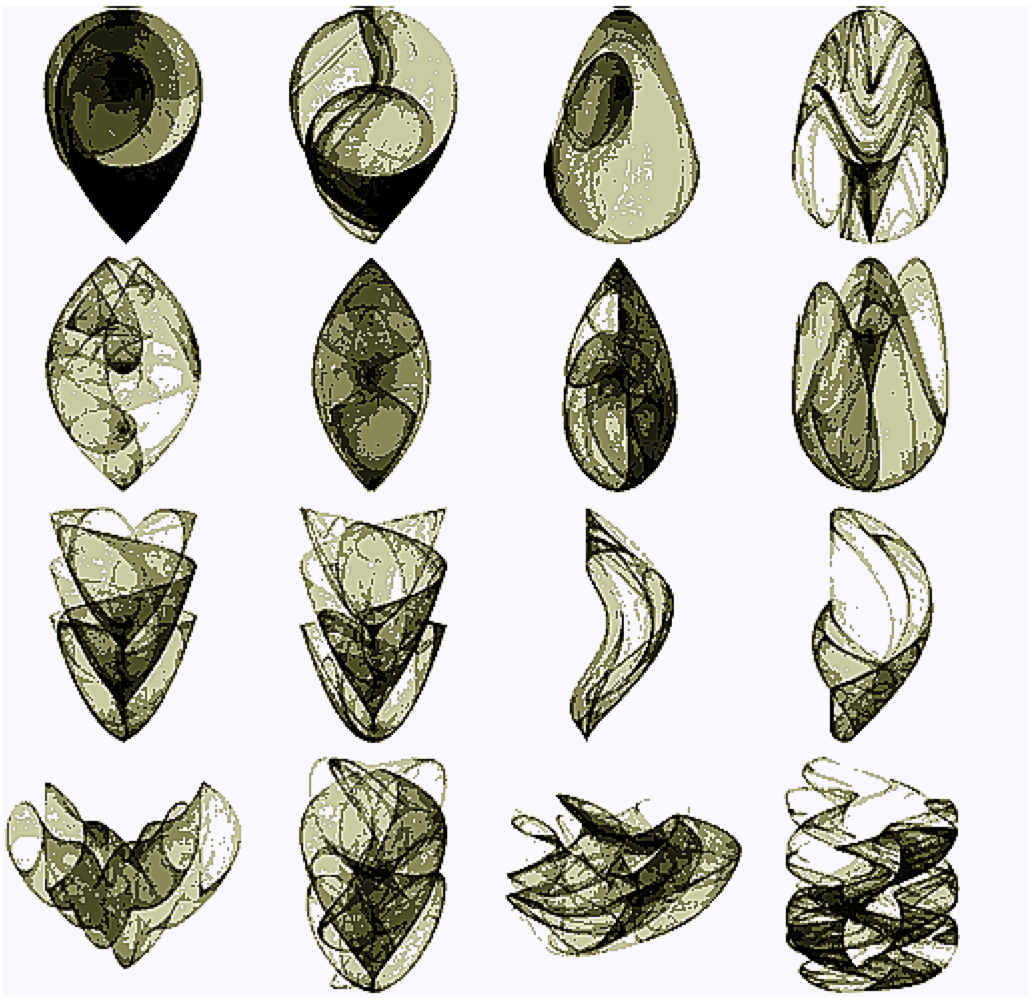


Fig. 3.41: Varieties of rendered strange attractors[1].

[1] Krawczyk, R.: *www.netcom.com/~bitart.*2002, Accessed 1/10/2007.

3-2 Spirolaterals

3-2-1 Introduction

3-2-1-1 Definition

Spirolaterals are mathematical figures that resemble fractals in its chaotic phenomena. The name Spirolateral is derived from two roots; lateral, referring to a flat surface, and Spiro, since the original series of spirolaterals was generated from the "square spiral". Spirolaterals are geometry with increasing length of turns and the turns repeating themselves with predefined angles.

They generate artistic two dimensional forms of unexpected complexity and beauty that might be used as ornaments or in any architectural organizational elements or for the purpose of inspiration and generation in the conceptual phase of design, before being guided, constrained, filtered and mutated by the utilitarian requirements and the functionalities of a building.

3-2-1-2 Outlines

Spirolaterals were encountered while investigating space curves and fractals. Researches uncovered what seems to be the first description of this geometrical form by Frank C. Odds[1], a British biochemist, Further research uncovered additional description by Abelson[2], Then Robert J. Krawczyk[3] find it a fertile area for generating unexpected infinite forms. Krawczyk focused on developing computer-based methods to investigate these two dimensional forms then begin to suggest three-dimensional versions of them, then the focus changed to investigate the mathematical rules of generating them that would result in visually more interesting designs.

[1] Odds, F.: *Spirolaterals*. Mathematics Teacher, pp.121-124, 1973.
[2] Abelson, H. and Disessa, A.: *Turtle Geometry*. MIT Press, pp.37-39, 120-122, 1968.
[3] Krawczyk, R.J.: *Spirolaterals, Complexity from Simplicity*. In International Society of Arts, Mathematics and Architecture, 1999.

3-2-1-3 The Mathematical Concept

The mathematical nature of spirolaterals based on the recursion of a predefined geometrical principles using mathematical and computer based tools. A spirolateral is created by drawing a set of lines; the first at a unit length, then each additional line increasing by one unit length while turning a constant direction. (Fig. 3.42) shows the systematic generation of an order 3 spirolateral, one that consists of 3 segments at turns of 90 degrees.

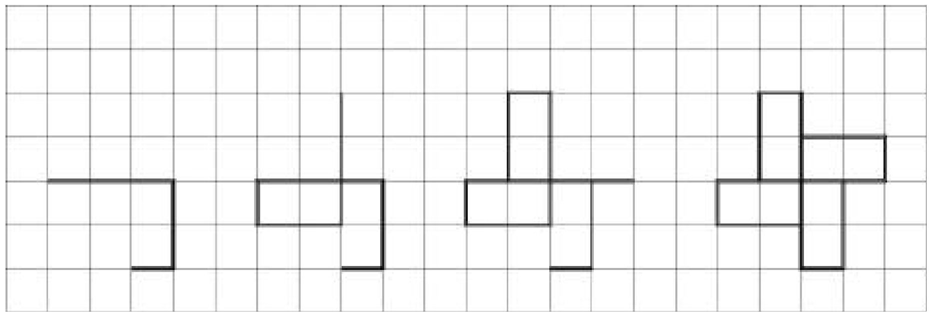


Fig. 3.42: Generation of an order 3 spirolateral.

Step 1: turn 90 degrees, draw one unit segment, turn 90 degrees, draw two unit segments, turn 90 degrees, draw three unit segments Step 2, 3 and 4: repeat Step 1. To complete a closed spirolateral, it is necessary to only to repeat the "square spiral" design, until the starting point is reached.

- **The Mathematical Properties of Spirolaterals**[1]

From the introductory example, the spirolateral is defined by three basic factors: the turning angle, the number of segments or turns, and the number of repetitions, which create a closed figure. To predict the closing property of a spirolateral, Krawczyk developed the following relationship. If the total turning angle is module of 360, then the spirolateral is closed.

Total turning angle = angle of turns x number of turns x repetitions.

[1] Ibid.

The exception is that when only one repetition become 360 degrees, the spirolateral will not be closed, or if repeated further it will just meander outward without ever closing on itself. Odds developed a notation to describe each of the spirolaterals. The number of turns, *n*, referred to as the "order" of the spirolateral, is the base number.

The angle of turn is written as a subscript; thus, 5_{90} would define a spirolateral with five turns each through 90 degrees. Odds suspected that any angle that is an exact divisor of 180 degrees would generate a spirolateral. Gardner[1] also agreed with Odds without investigating any other possible angles. (Fig. 3.43) demonstrates closed spirolaterals based on an angle of 180/*n*.

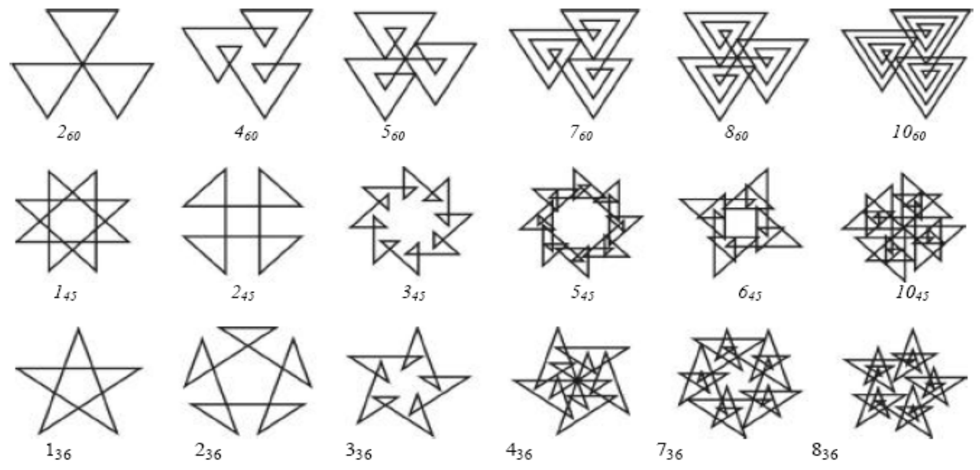


Fig. 3.43: Spirolaterals based on an angle 180/n.

Odds then introduced the concept that not all the turns need to be in the same direction. For any series of segments, some of the turns can be to the right and others can be to the left. The notation he suggests is: $7_{90\ 4,6}$ for a spirolateral of 7 turns at 90 degrees in which the 4th and 6th turns are in the opposite direction. (Fig. 3.44) shows a series of reverse turn spirolaterals based on the angle of 90 degrees. For a specific order *n* there are 2_n possible

[1] Gardner, M.: *Knotted Doughnuts and Other Mathematical Entertainments*. W. H. Freemand and Company, pp. 205-208, 1986.

turns that generate spirolaterals, half of those are mirror images; so 2_{n-1} unique figures are possible in each order. From the analysis of the closure relationship previously described it has been found that spirolaterals, which close for all turns in the same direction, will also close for any reversed turns given the same number of turns and repetitions[1].

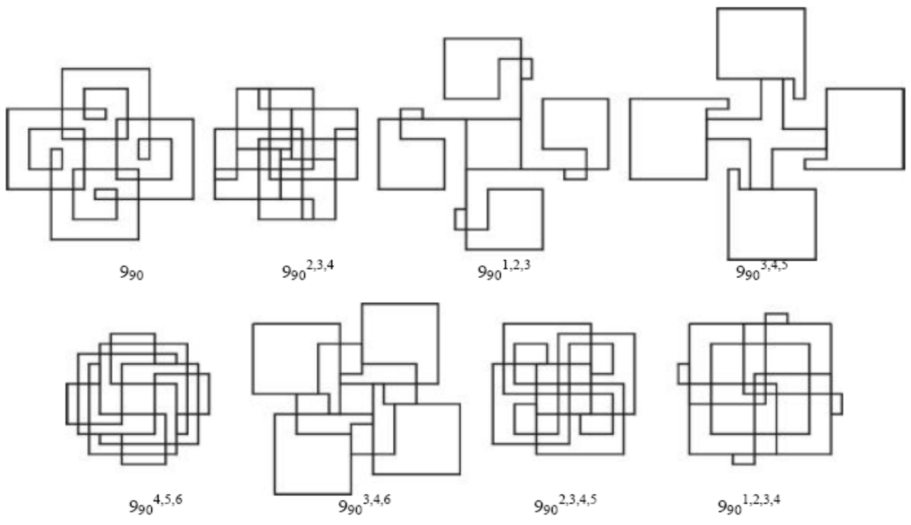


Fig. 3.44: A 9_{90} Spirolaterals with reverse turns.

Another example is given in (Fig. 3.45) of a 6_{45} spirolateral.

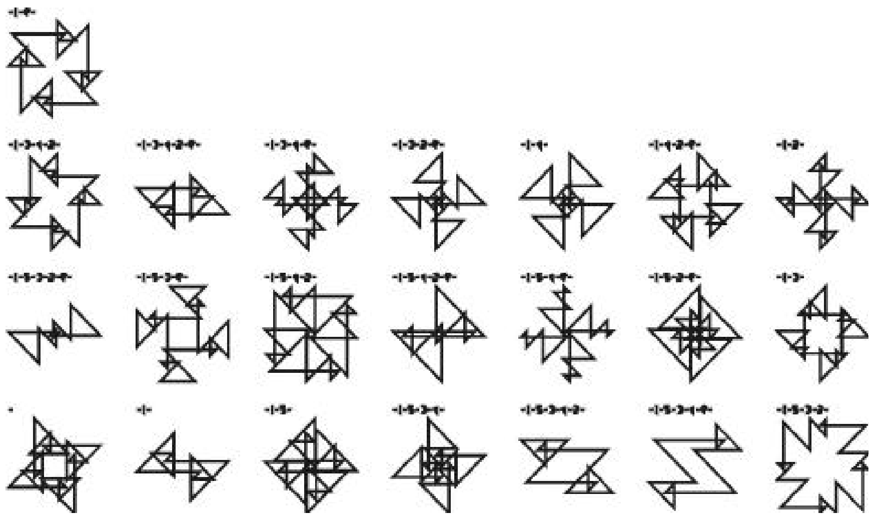


Fig. 3.45: A 6_{45} Spirolateral with only unique reverse turns.

[1] Krawczyk, R.J.: *The Art of Spirolateral Reversal*. The International Society of Arts, Mathematics and Architecture, edited by N. Friedman, University of Albany-SUNY, 2000.

3-2-2 Classifications

Spirolaterals classified into two main types: straight and curved.

3-2-2-1 Straight Spirolaterals

The first studies on spirolaterals didn't break away from the canon of straight lined spirolaterals. The first applications simply used a single line to represent each spirolateral. Visually it was very weak. Robert J. Krawczyk added a line thickness to it. The forms turned from simple line drawings to iconic symbols. As the line thickness was increased, the open areas between the lines began to close and a totally different form emerged (Fig. 3.46 a).

In addition to displaying the line thickness, options are included to add the centerline and edge lines or both for each. (Fig. 3.46 b, 3.46 c and 3.46 d) display these variations. These options displayed the internal geometry of the spirolateral that was somewhat lost to the line thickness[1].

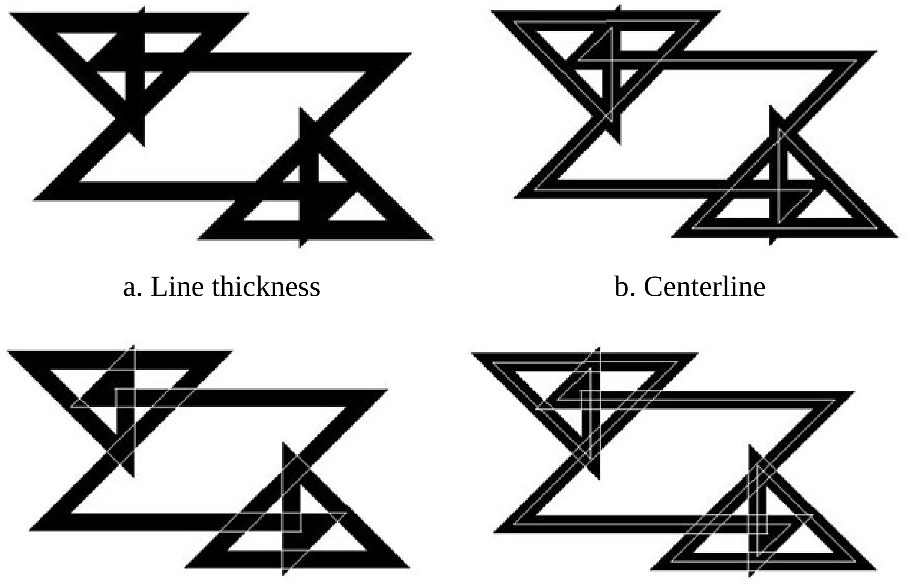


Fig. 3.46: Variation of thickness and lines.

[1] Ibid.

(Fig. 3.47) display selected examples from JAVA spirolateral-generating module. (Fig. 3.48, 3.49) display two types of spirolaterals closed and closed with reversals[1].

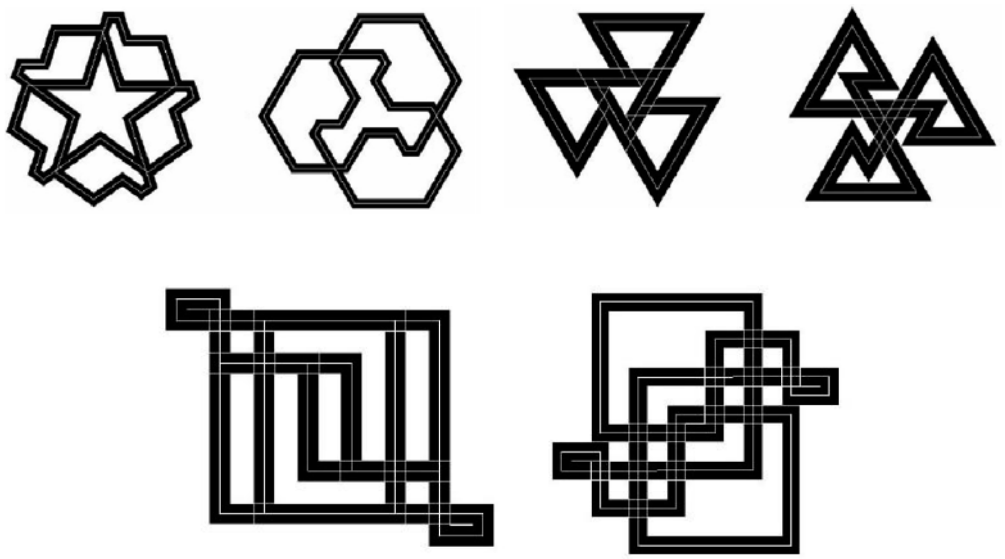


Fig. 3.47: Selected examples.

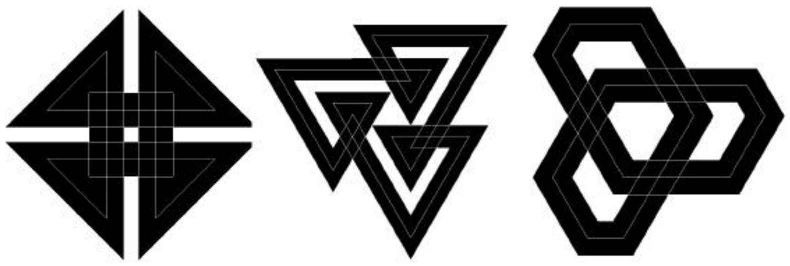


Fig. 3.48: Closed spirolaterals.

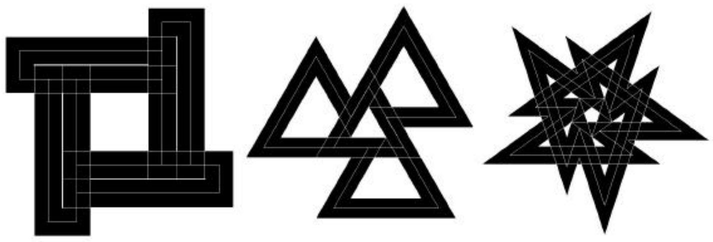


Fig. 3.49: Closed spirolaterals with reversals.

[1] Ibid.

3-2-2-2 Curved Spirolaterals

Investigations had been made to transform the straight spirolaterals into curves, a series of artworks developed showing possible curving methods: curve fitting, spins, inversion, antimercator and circular transformations, along with hypocycloid and epicycloids curves[1].

A spirolateral consists of lines drawn between two vertices. Instead of using the actual lines, one idea was to use only the vertices. Using CAD-based software, procedures were written to interpret the vertices as control points for curves. (Fig. 3.50) displays this concept. The simple 1_{90} spirolateral, a square, consists of four vertices. These vertices can be fitted with arcs at the start and end of adjacent vertices. The vertices can also be control points to generate a spline curve.

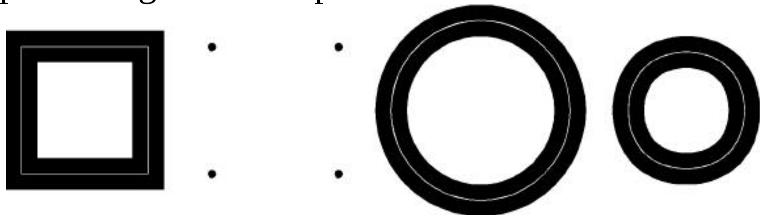


Fig. 3.50: Curved spirolateral 1_{90}.

Many of the spirolaterals developed as curves were found to be interesting while some not visually exciting nor did they generate any unexpected images. The first set created were ones that consisted of ten or less turns (Fig. 3.51).

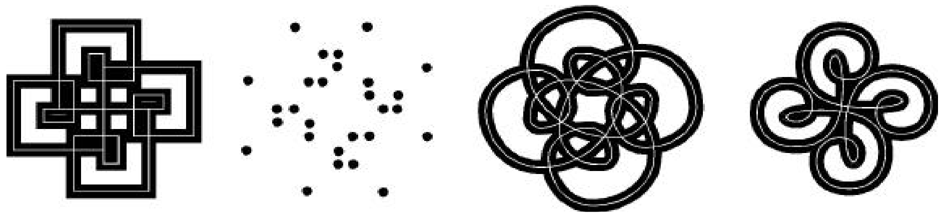


Fig. 3.51: Curved spirolateral 7_{90}.

[1] Krawczyk, R.J.: *More Curved Spirolaterals*. College of Architecture, Illinois Institute of Technology, USA. BRIDGES: Mathematical Connections In Art, Music, and Science, 2001

Only when the number of turns increased to a range of 11 to 15, with reversed turns included, did the number of control points increase to the point where they begin to generate images of unexpected detail and complexity. (Fig. 3.52) displays two such examples.

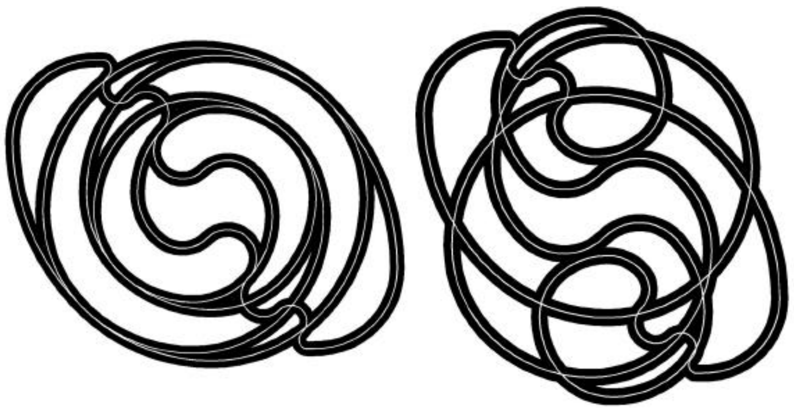


Fig. 3.52: Curved spirolaterals of 14_{90}.

- **Curved Spirolaterals Types**

Many approaches to curve spirolaterals had been developed like applying a transformation, a projection, or mapping from a linear image to a curved one.

a- Curves by Inversion

In researching these possibilities, the first one encountered was the concept of inversion, as described by Weisstein[1], Dixon[2] and Lawrence[3]. This mathematical method turns lines into circles. The transformation based on the image midpoint is as follows:

$X = (x * r_2) / (x_2 + y_2)$

$Y = (y * r_2) / (x_2 + y_2)$

(Fig. 3.53) demonstrates this transformation on a simple square spirolateral. Inversion by definition reconstructs the spirolateral as curves. After transforming a variety of spirolaterals, it was found that ones that are open in the center generate the most interesting results. Ones that have lines that cross the image midpoint

[1] Weisstein, E.W.: *Concise Encyclopedia of Mathematics.* CD-ROM edition 1.0, Chapman & Hall/CRCnetBASE, 1999.
[2] Dixon, R.: *Mathographics*. Dover Publications, Inc., p.152, 1987.
[3] Lawrence, J.D.: *A Catalog of Special Curves*. Dover Publications, Inc., p.43, 1972.

generate very strange results, which are visually not interesting. Symmetry also added to the quality of the results, as did the selection of the line thickness. (Fig. 3.54) displays a sample of spirolateral inversions.

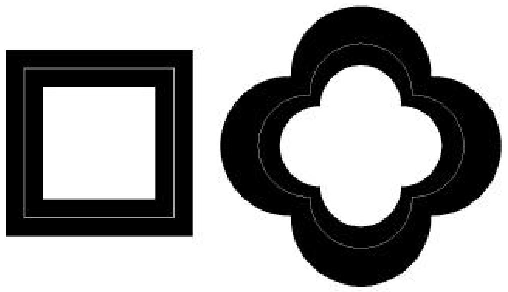


Fig. 3.53: Inversion of spirolateral 1_{90}.

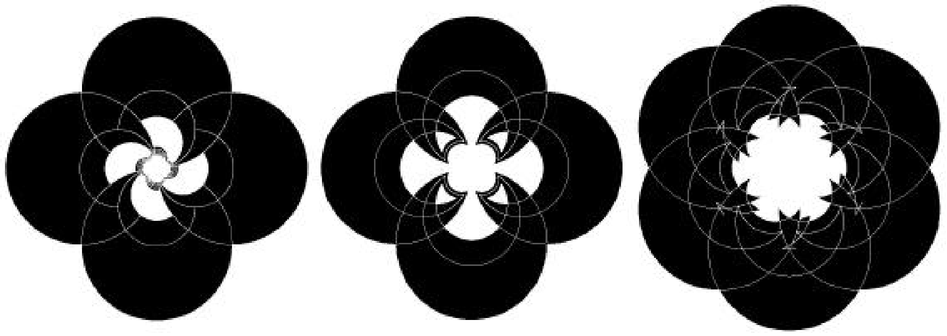


Fig. 3.54: Spirolateral inversions.

b- **Circular Transformations**[1]

Continuing the search for other methods of curving spirolaterals, Dixon outlined a method he calls anitMercator. Horizontal lines become circles concentric with the coordinate origin, vertical lines become radial, and slanting lines become logarithmic spirals. The transformation in polar coordinates is as follows:

A = k * x where: k = 2π/(xmax-xmin)

R = exp (k * y)

(Fig. 3.55) demonstrates this transformation on a simple square spirolateral. This transformation is affected by the line thickness and also by the offset from the origin. The origin is positioned in the lower-left corner of the image. This location results in the

[1] Krawczyk, R.J.: *More Curved Spirolaterals*. College of Architecture, Illinois Institute of Technology, USA. BRIDGES: Mathematical Connections in Art, Music, and Science, 2001, op.cit.

image being bent clockwise starting with the left corner. (Fig.3.56) displays a sample of spirolateral transformed by antiMercator.

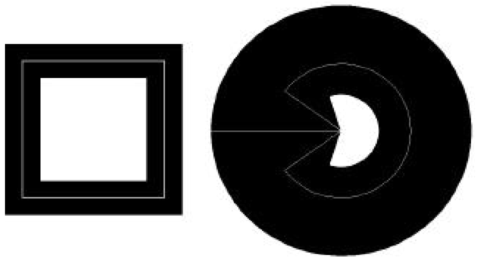


Fig. 3.55: antiMercator spirolateral 1_{90}.

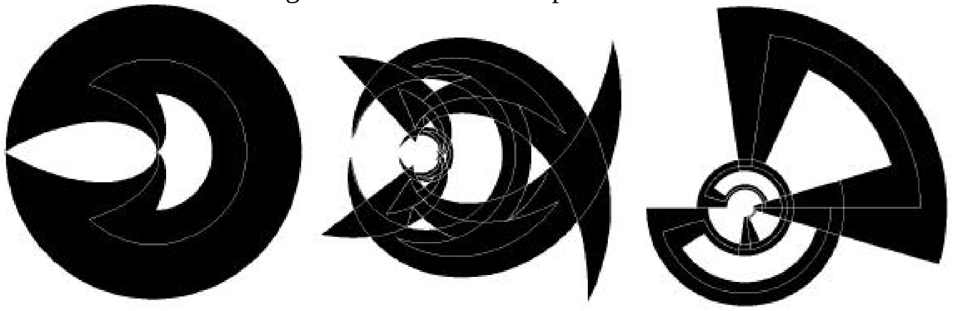


Fig. 3.56: antiMercator spirolaterals.

While investigating the antiMercator transformation, one alternate method was found by removing the exponential function. The circular form remains without the logarithmic spiral effect. Since no formal name has been found for this transformation, it will be referred as simply circular. (Fig. 3.57) demonstrates this circular transformation on a simple square spirolateral. This transformation differs from the antiMercator in that the line spacing is of a more constant distance from the center, so that the original distances are better represented[1]. This transformation also changes as the offset increases. (Fig. 3.58) displays a sample of spirolateral transformed by the Circular transformation.

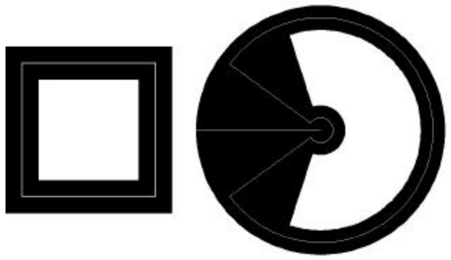


Fig. 3.57: Circular spirolateral 1_{90}.

[1] Ibid.

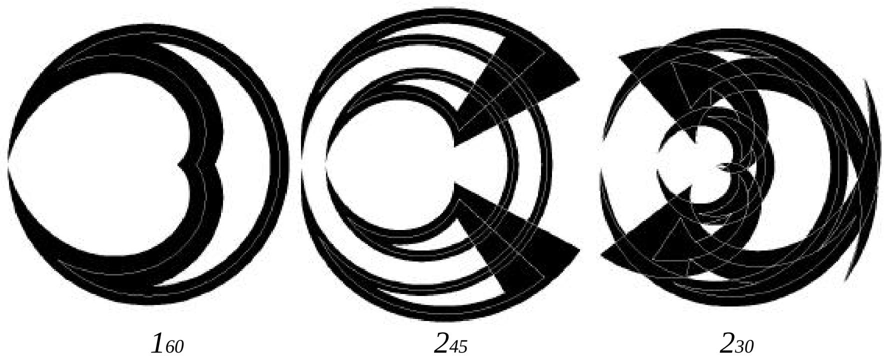


Fig. 3.58: Circular spirolaterals.

c- Following a Curve[1]

The antiMercator and Circular transformation consist of computing polar coordinates based on the original image and then converting them to Cartesian coordinates. An alternate method is to not converting the polar to rectangular Cartesian coordinates but to follow curved ones. Using the angle and radius computation from the Circular transformation, the modified Cartesian coordinates are computed with the following curves, as in Lawerence:

Hypocycloid: X = R * (cos (A)*(n-1) + cos (A*(n-1))) where: n = number of cusps, 3 and 4. Y = R * (sin (A)*(n-1) – sin (A*(n-1)))

Epicycloid: X = R * (cos (A)*(n+1) – cos (A*(n+1))) where: n = number of cusps, 2, 3, and 4. Y = R * (sin (A) (n+1) – sin (A*(n+1)))

The major difference between these two curves is that the Hypocycloid produces concave curves and the Epicycloid produces convex ones. (Fig. 3.59, 3.60) demonstrate these transformations on a simple square spirolateral.

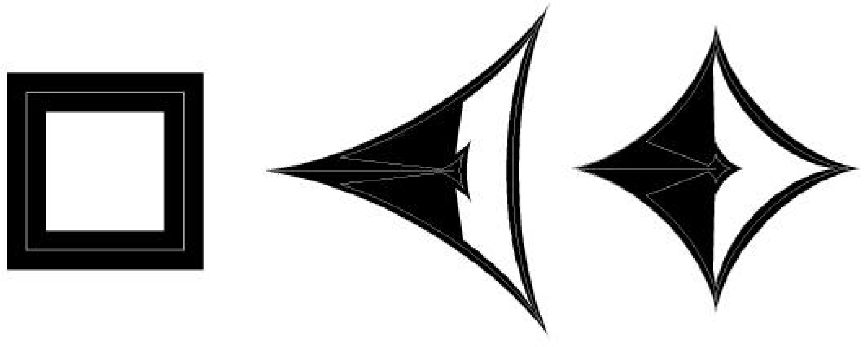


Fig. 3.59: Hypocycloid spirolateral 1_{90}.

[1] Ibid.

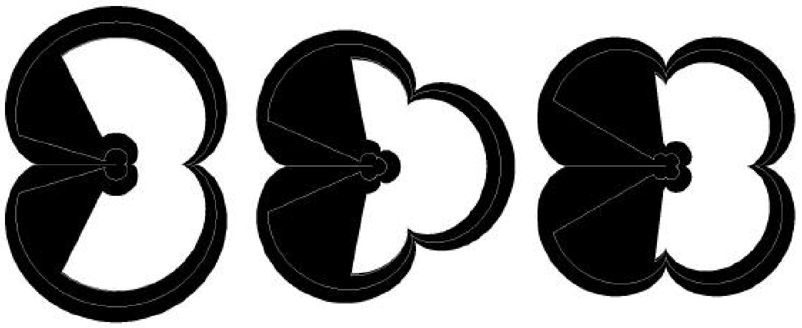
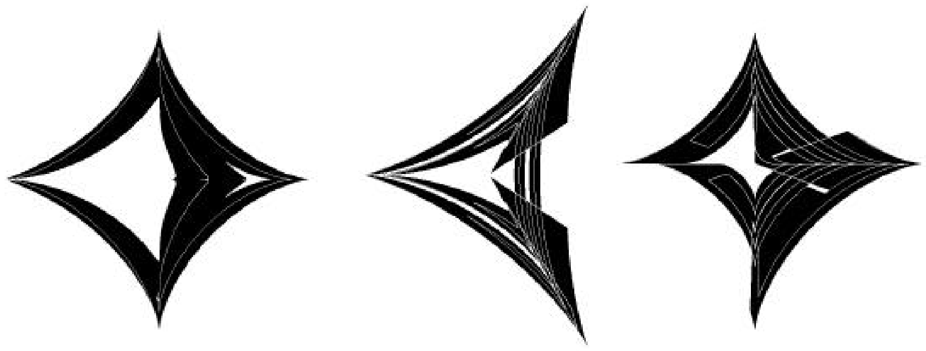


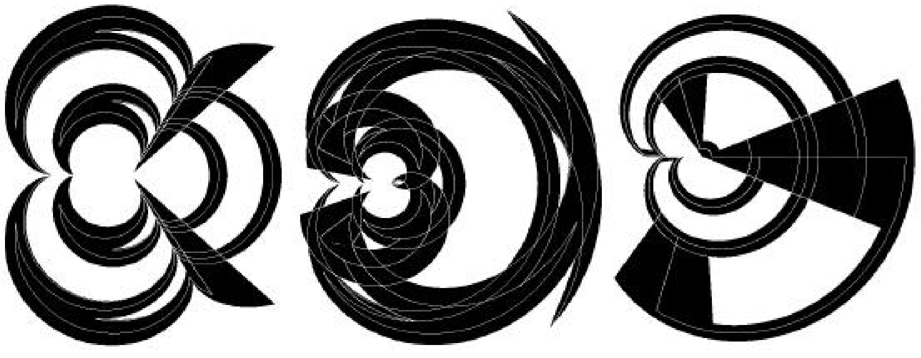
Fig. 3.60: Epicycloid spirolateral 1_{90}.

(Fig. 3.61, 3.62) display sample of spirolaterals using the Hypocycloid and Epicycloid curves.



1_{60} 2_{45} 3_{90}

Fig. 3.61: Hypocycloid spirolaterals.



2_{45} 2_{30} 3_{90}

Fig. 3.62: Epicycloid spirolaterals.

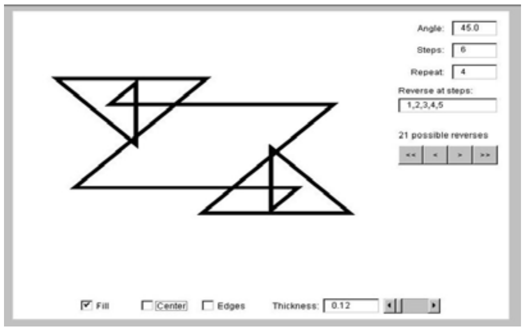
These transformations produce more delicate curves and unexpected results due to the effect of the line thickness.

3-2-3 Architectural Potentials

3-2-3-1 Generating 2D Forms

A number of different attempts have been made to use the forms generated by the spirolaterals in two-dimensions to create actual artwork, could be used as masses or master plans or landscaping or tiling or any other architectural organizational elements.

A series of programs have been written to generate spirolaterals. The first one written in Microsoft QuickBASIC for computational inquiry and AutoCAD's AutoLISP for visualization. The first version simply used a single line to represent each spirolateral. Visually it was very weak. A line thickness was added. The forms turned from simple line drawings to iconic symbols. As the line thickness was increased, the open areas between the lines began to close and a totally different form emerged. Once the basics were understood a JAVA version (Fig. 3.63) was written to display the spirolaterals[1].


Angle: 45.0
Steps: 6
Repeat: 4
Reverse at steps: 1,2,3,4,5
21 possible reverses

☑ Fill ☐ Center ☐ Edges Thickness: 0.12

Fig. 3.63: The JAVA version.

(Fig. 3.64) to (Fig. 3.76)[2] show 13 galleries of different types of spirolaterals , straight and curved that could be generated using a JAVA application by just varying 8 parameters in the JAVA interface.

These generated patterns strongly aid the preliminary design phase, empower creativeness, trigger inventiveness and could be an initiator for a three dimensional composition.

[1] Krawczyk, R.J.: *The Art of Spirolateral Reversal*. The International Society of Arts, Mathematics and Architecture, edited by N. Friedman, University of Albany-SUNY, 2000. op.cit.

[2] Krawczyk, R.J.: *www.netcom.com/~bitart.*2002. Accessed 1/10/2007. op.cit.

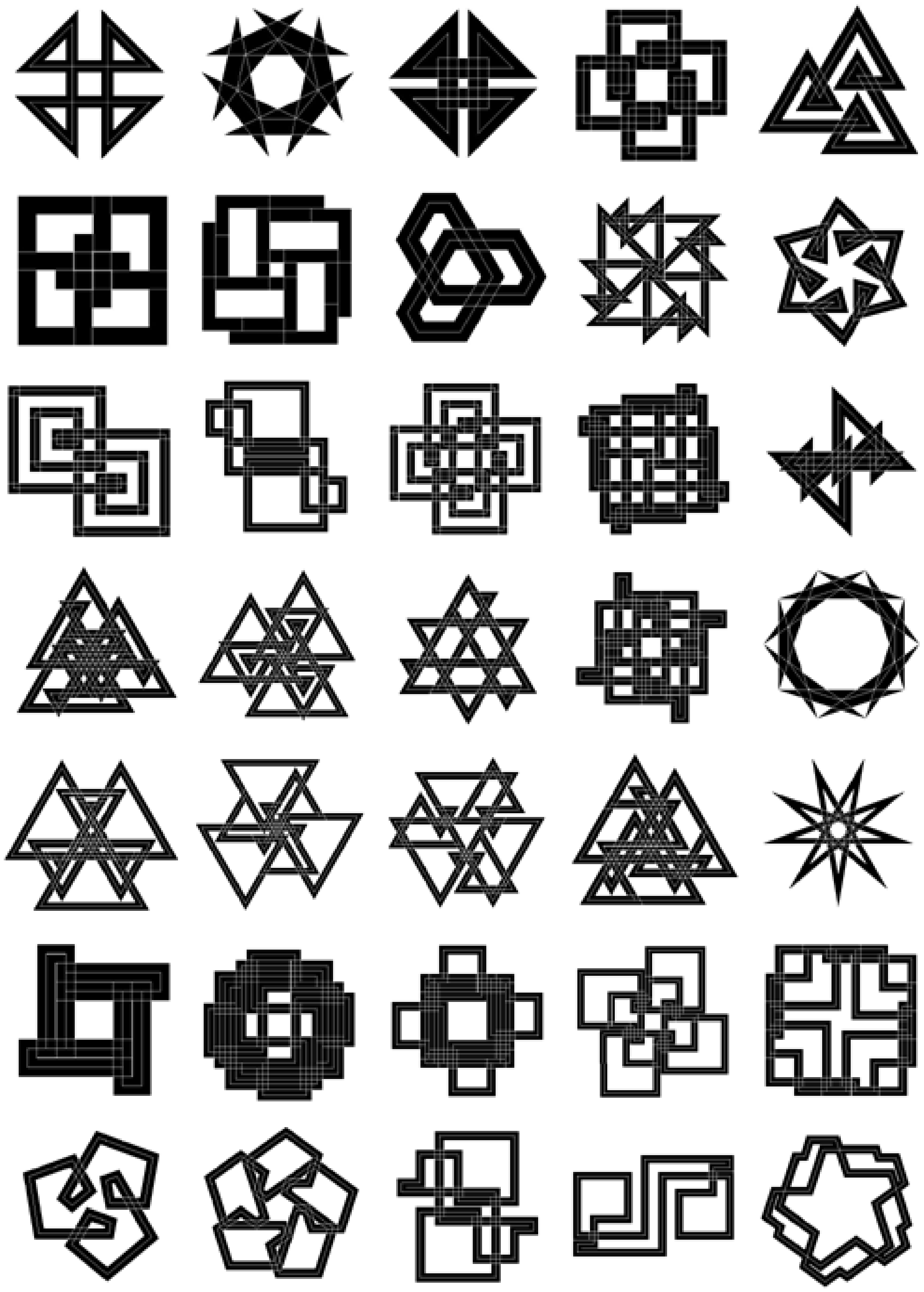


Fig. 3.64: Straight spirolaterals series.

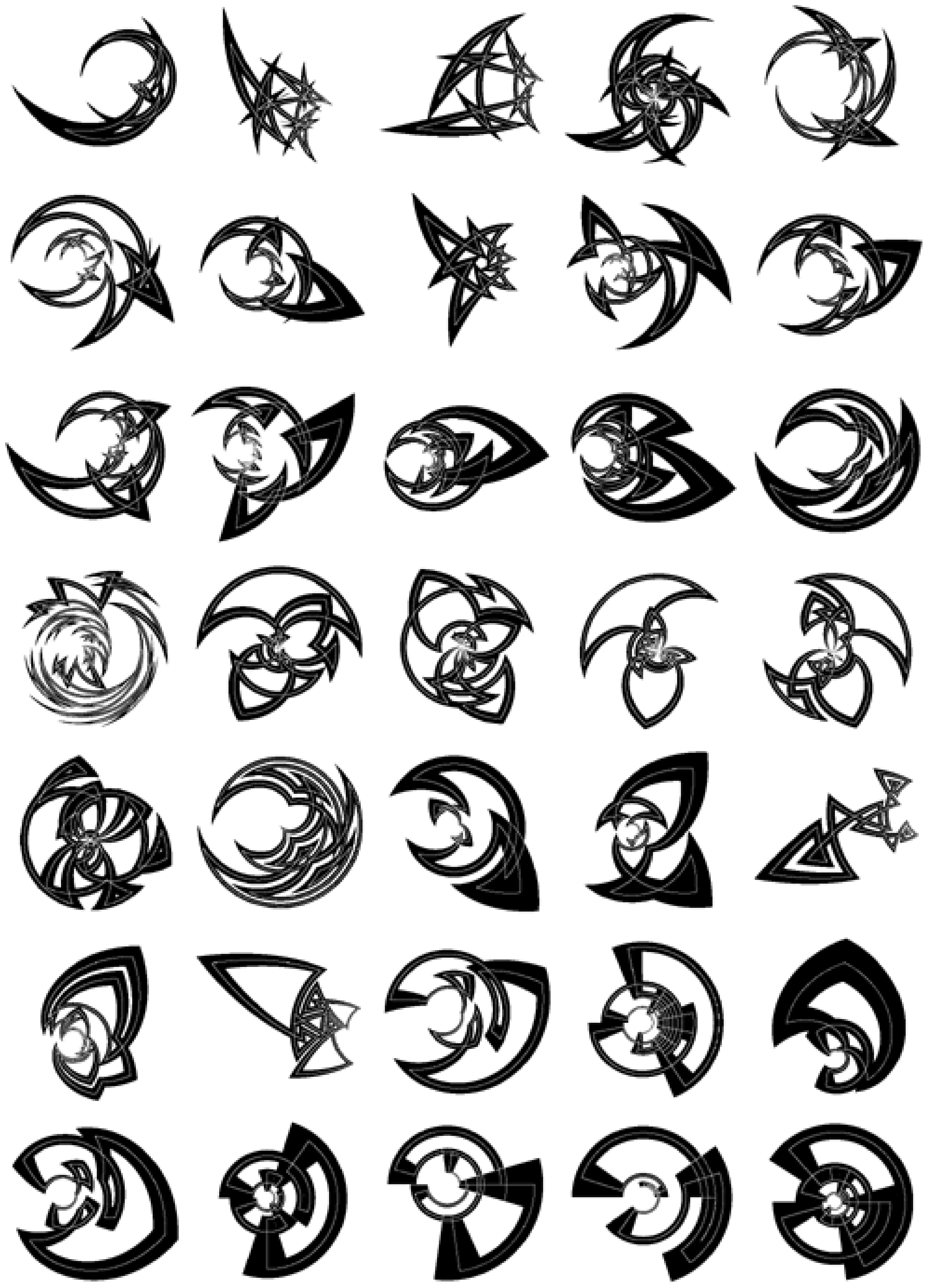


Fig. 3.65: Circular & antiMercator series.

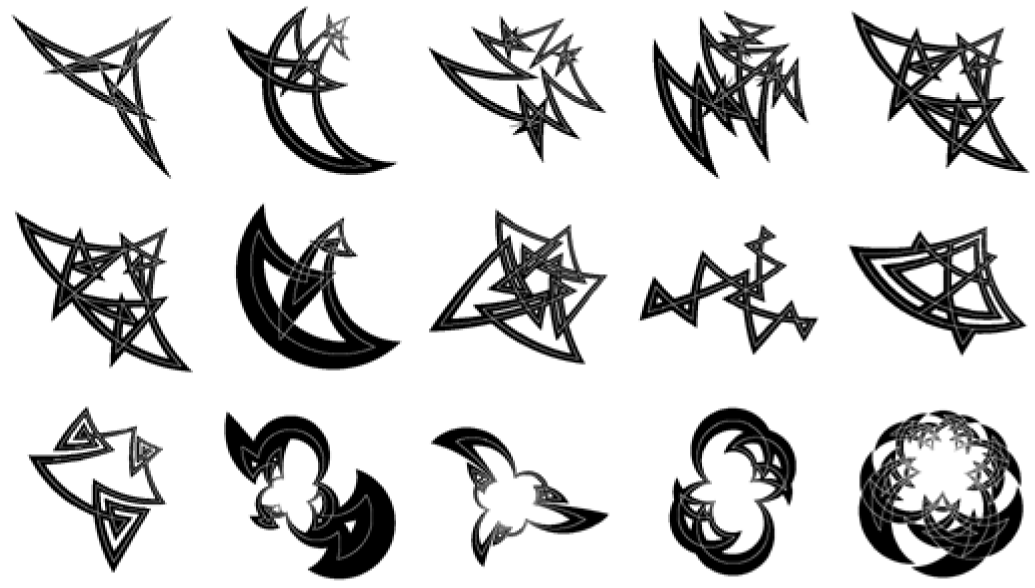


Fig. 3.66: Inversion series.

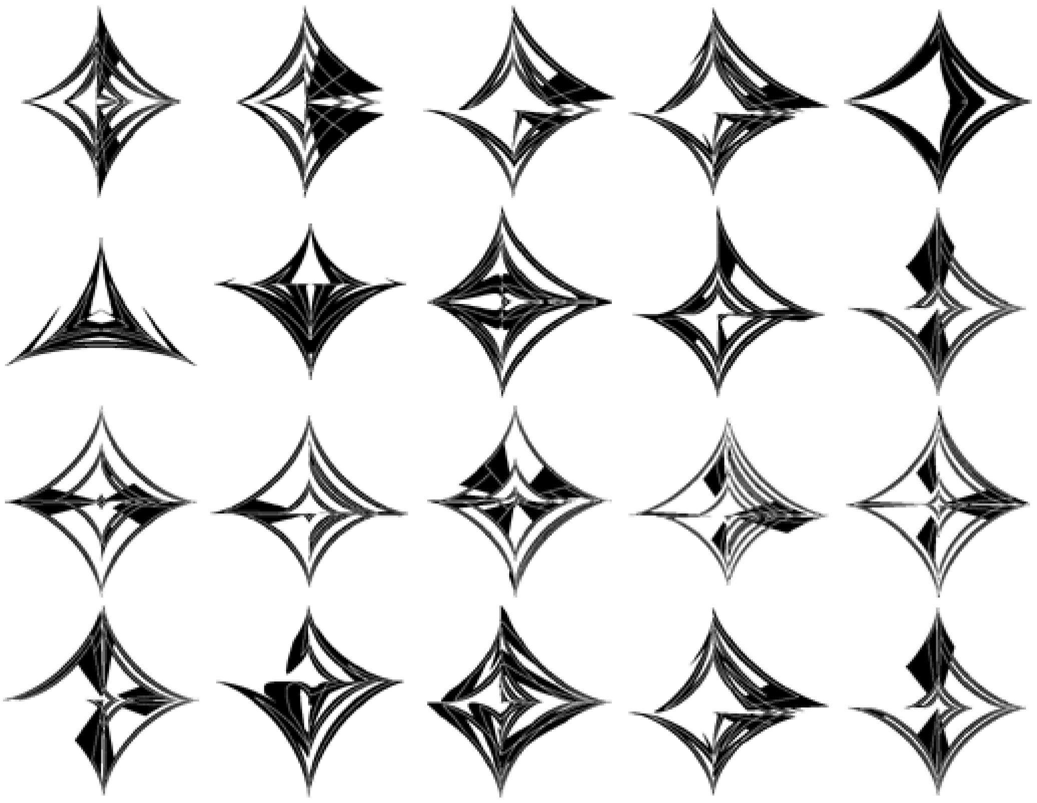


Fig. 3.67: Hypocycloid series.

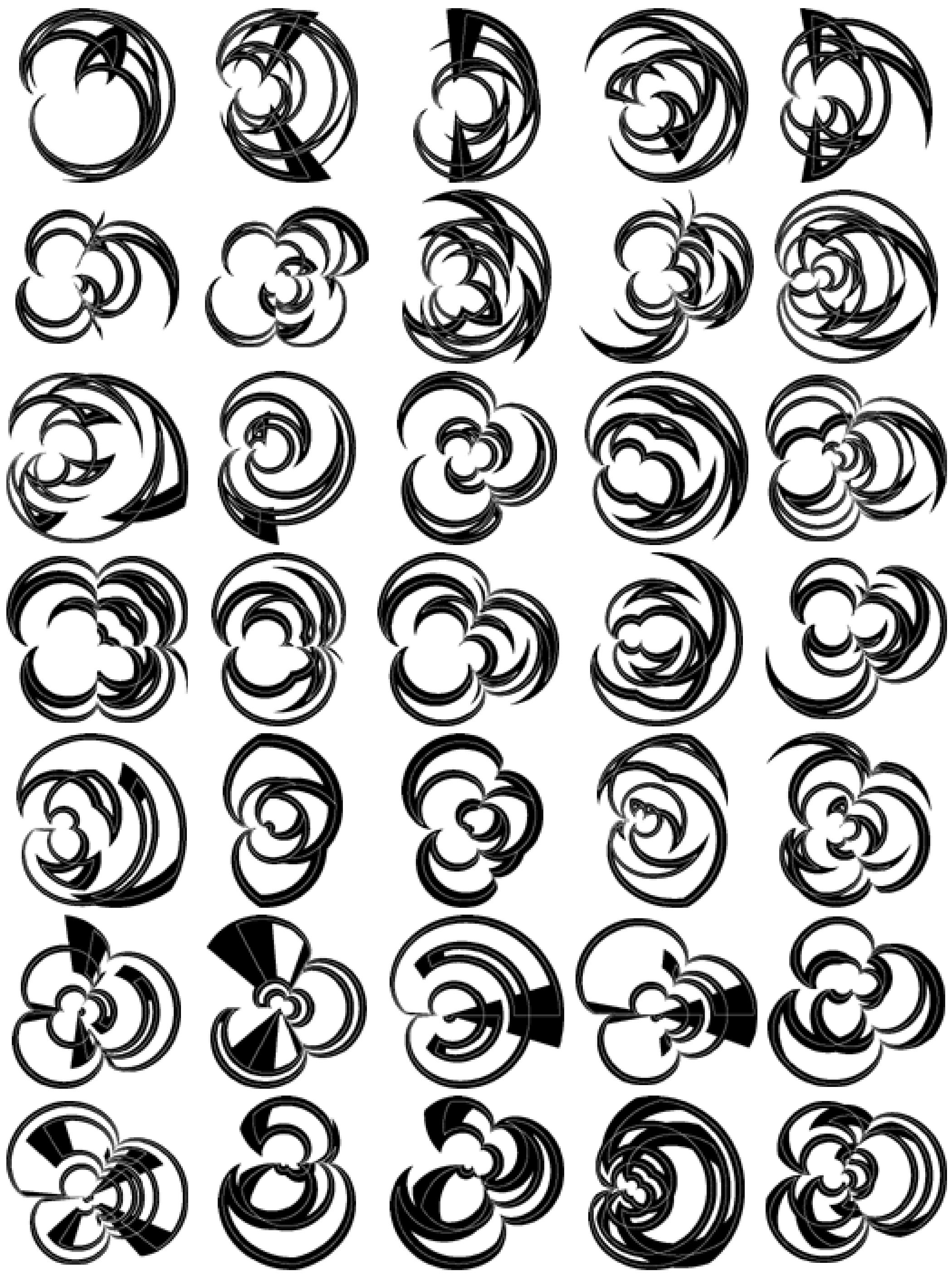


Fig. 3.68: Epicycloid series.

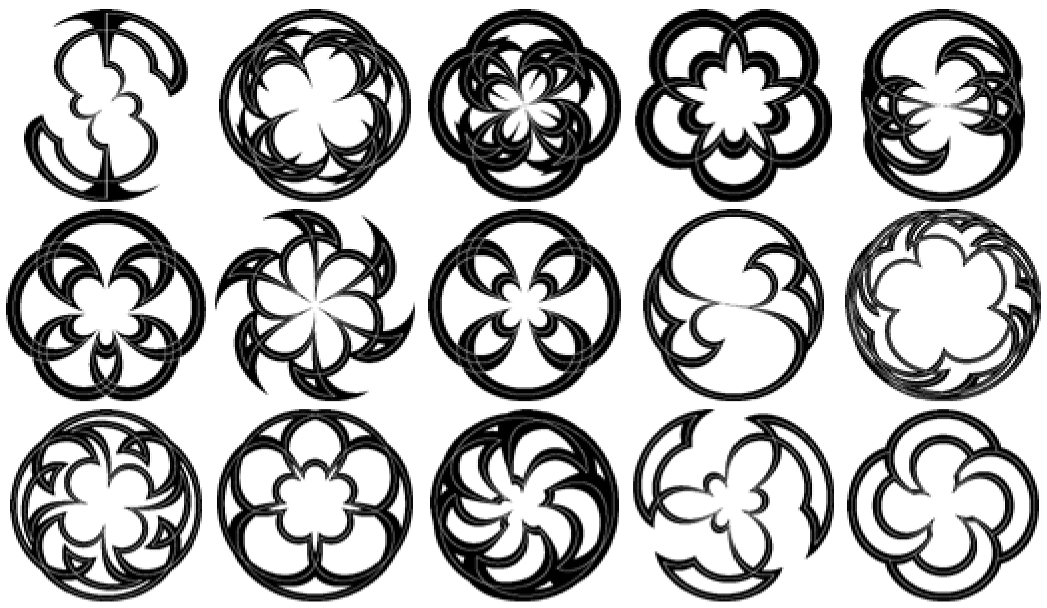


Fig. 3.69: Harmonic Mean Inversion series.

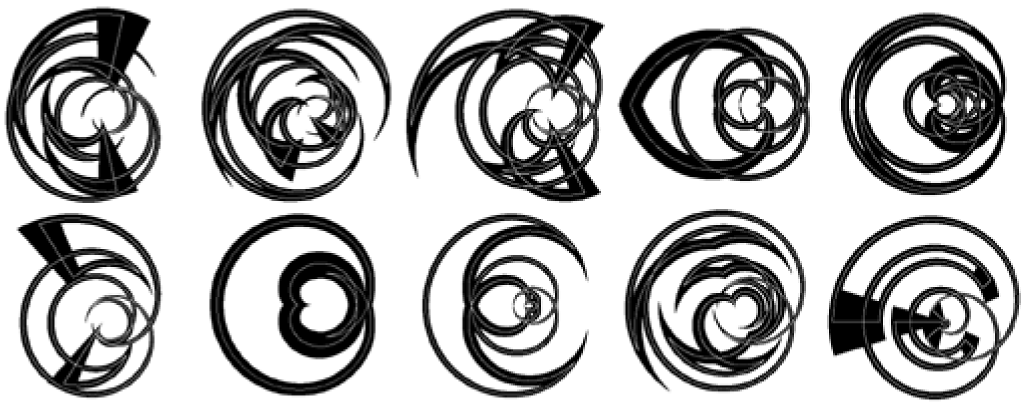


Fig. 3.70: Epitrochoid Curve series.

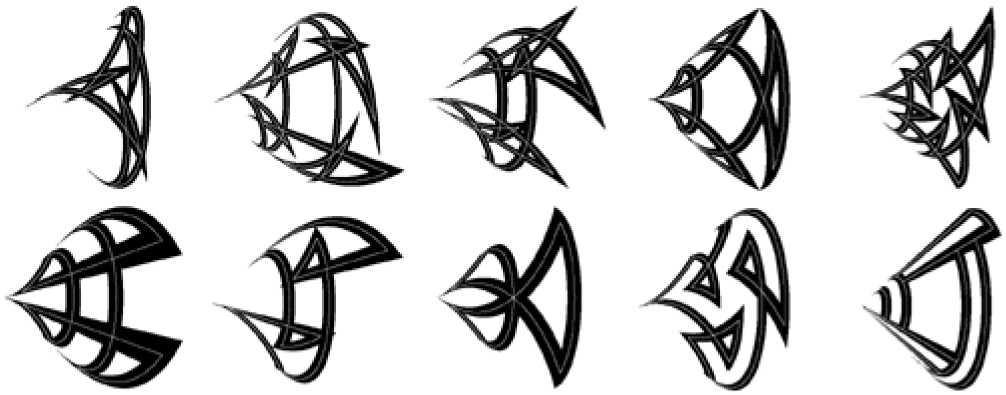


Fig. 3.71: Piriform Curve series.

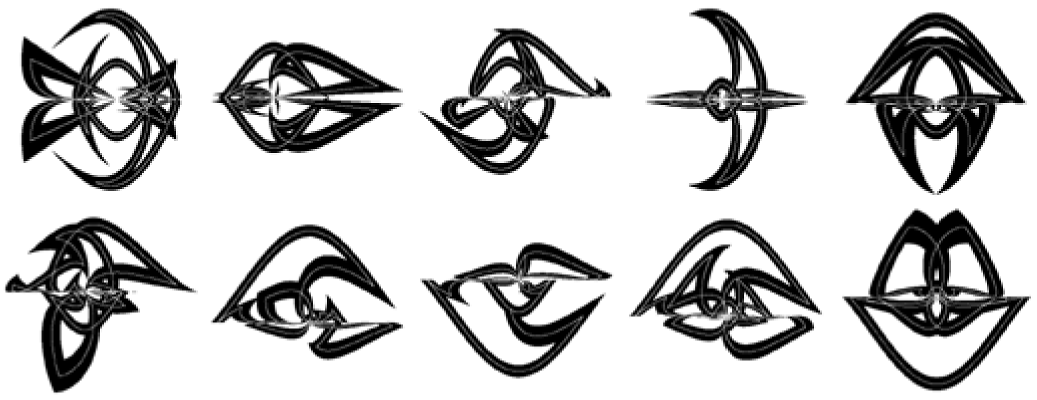


Fig. 3.72: Bicorn Curve series.

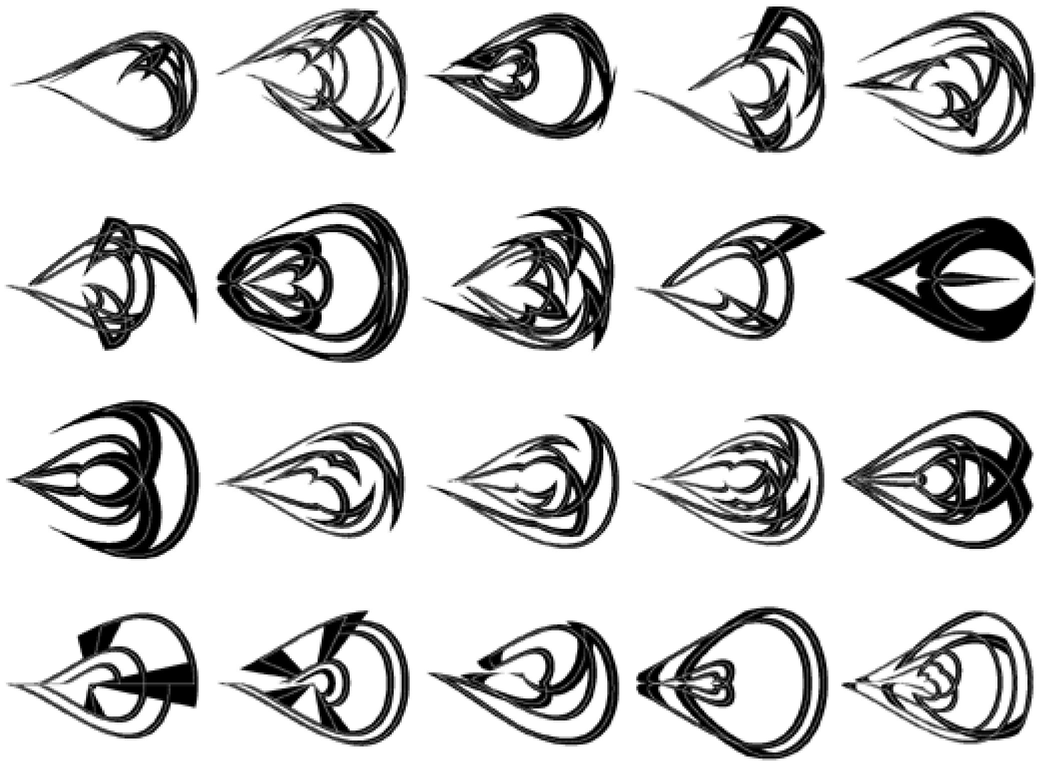


Fig. 3.73: Tear Drop Curve series.

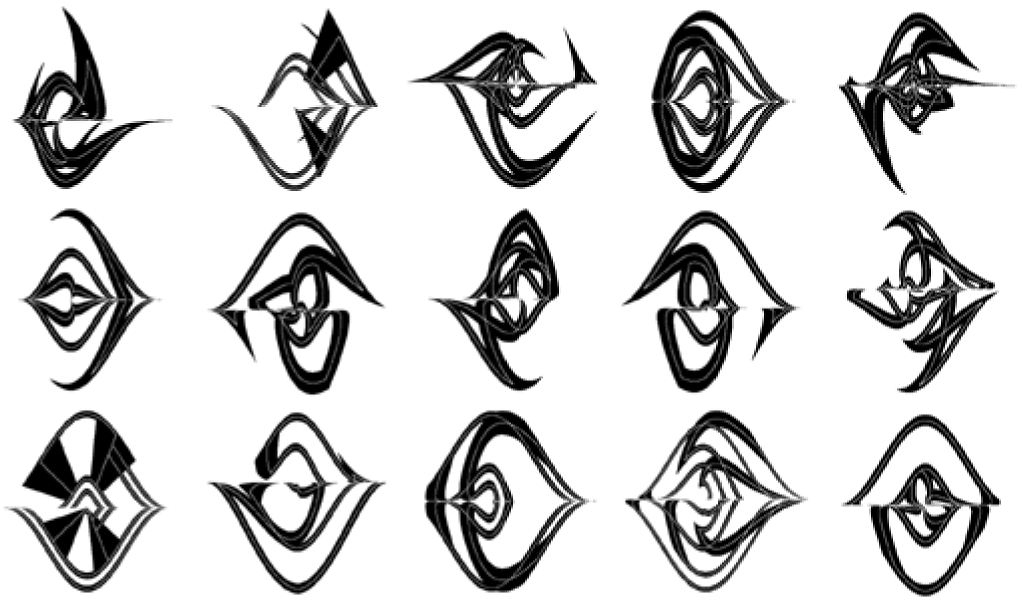


Fig. 3.74: Lips Curve series.

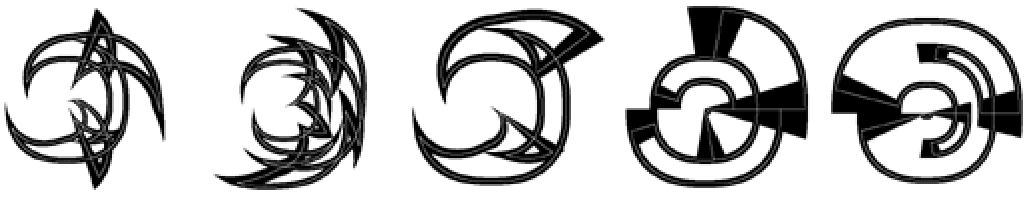


Fig. 3.75: Lame Curve series.

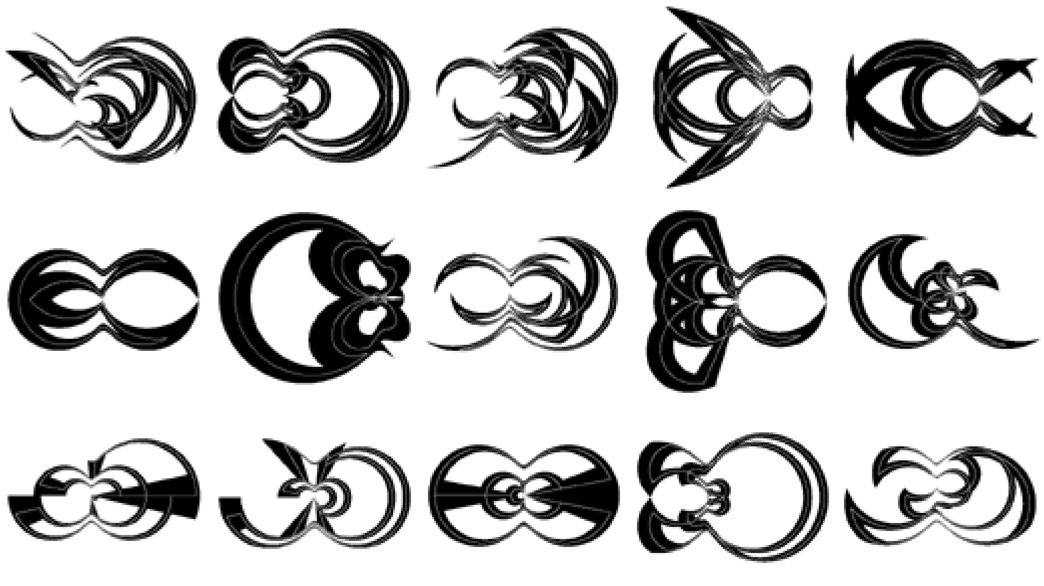


Fig. 3.76: Hippopede Curve series.

3-2-3-2 Generating 3D Forms

All the previous investigations have created spirolaterals in two-dimensions. For this particular, series of three-dimensional constructions were investigated. (Fig. 3.77) displays the spirolateral that was selected. (Fig. 3.77 a, 3.77 b) are the original spirolateral, and (Fig. 3.77 c) is the variation selected. In this version the line thickness is decreased so to better articulate the turns as separate sculptural elements.

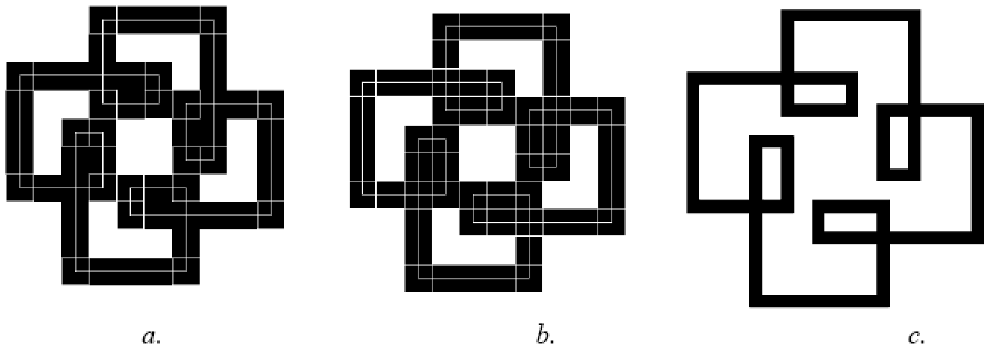


Fig. 3.77: A 4_{90} spirolateral to be transformed to a 3d composition.

In moving these figures into the third dimension, an interpretive approach was examined. This approach attempts to use every possible property of the spirolateral and its potential to define the third dimension. The overall concept was to follow the given geometry, retain the step and turn properties of the original spirolateral, and retain the entire form. Three basic approaches were explored; as a relief, as an assembly, and as a construction[1].

- **As a Relief**

The first interpretation was as a relief, a simple three-dimensional extrusion of the two-dimensional figure. The relief concept continues the line quality of the original spirolaterals in three dimensions. (Fig. 3.78 a) displays the simplest extrusion that is possible. All of the parts of the spirolateral have the same

[1] Krawczyk, R.J.: *Sculptural Interpretation of a Mathematical Form.* College of Architecture, Illinois Institute of Technology, USA, Bridges, Towson University, 2002.

dimension and the same height. The extrusion can also be viewed in a positive and negative fashion, (Fig. 3.78 b) displays the negative and (Fig. 3.78 c) displays a combination of the positive and negative.

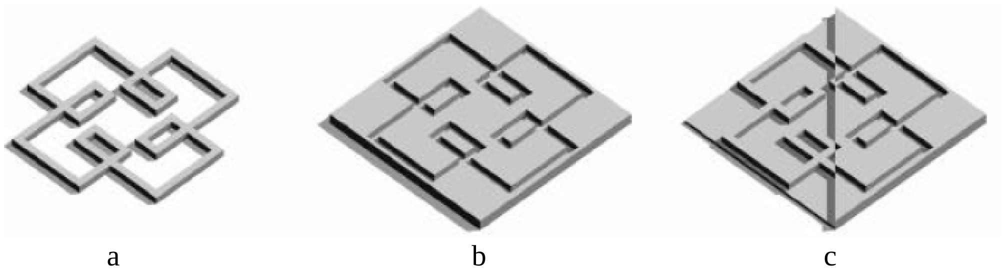


Fig. 3.78: Simple relief, positive and negative.

Continuing with the relief concept, the next sets of pieces consider the spirolateral property of turn size and its change in length. First considered is each turn individually, (Fig. 3.79 a) increases the height of each individual turn from its starting turn to the ending turn, a total of 20 turns. The height parallels the change in turn length. A simple variation would be to reverse the turn size from increasing to decreasing from the starting turn. Another interpretation of the increasing turn size is to combine the increasing and decreasing turn height. (Fig. 3.79 b) increases each turn height until the midpoint is reached, then the turn height is decreased[1].

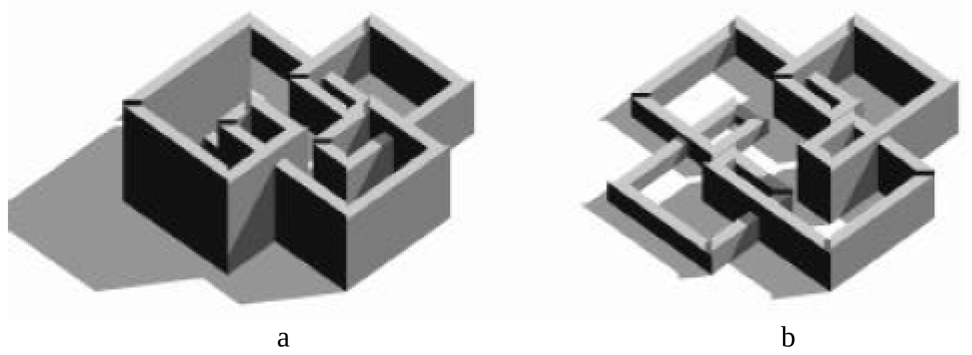
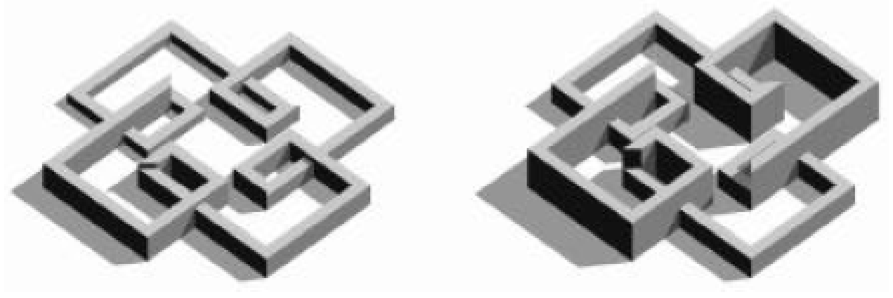


Fig. 3.79: Increasing/decreasing turn height.

The next consideration was treating each set of turns as a unit, not as individual turns. (Fig. 3.80 a) increasing the height of each set

[1] Ibid.

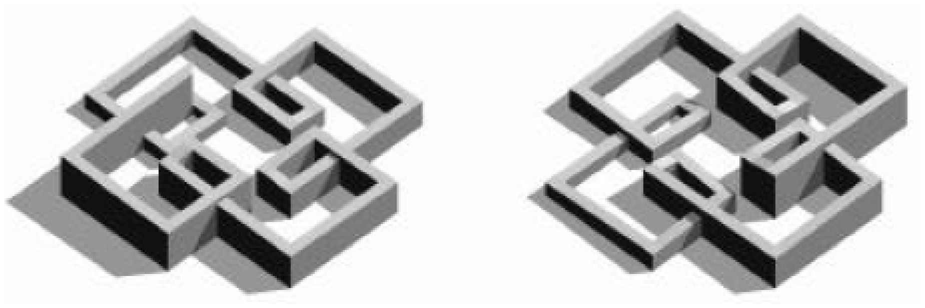
of turns, in this case, there are four set of turns. A simple variation would be to reverse the pattern of height from increasing to decreasing. Another variation is to incorporate both increasing and decreasing. (Fig. 3.80 b) alternates the height of each set of turns.



a b

Fig. 3.80: Increasing/decreasing set of turn height.

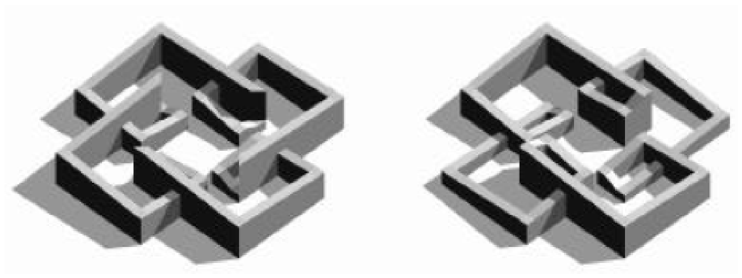
Using the same type of variations as in the previous pieces, stepping the turns, a series of variations was considered that ramped the turns. (Fig. 3.81 a) simply ramps the turn height from the starting turn to the ending. A variation would be to reverse the increasing ramp to decreasing. (Fig. 3.81 b) increases the ramp from the starting turn to the midpoint and then decreases it to the final turn. It can also be varied by revering the increase and decrease direction.



a b

Fig. 3.81: Increasing/decreasing turn height as ramps.

As with the stepped variations, the ramping can also consider an entire set of turns not just each individual one. (Fig. 3.82 a) increases the ramp for each set of turns and then resets the ramping to start it over again. (Fig. 3.82 b) alternates the increasing and decreasing of the ramp for each set of steps. As before a reverse variation is also possible for both.

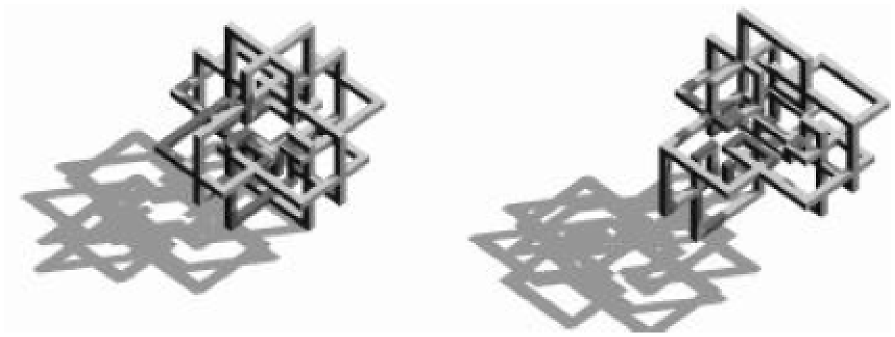


a b

Fig. 3.82: Increasing/decreasing set of turns height as a ramp.

- **As an Assembly**[1]

A simple assembly can also be created by taking the entire spirolateral and comibidbining it with itself. (Fig. 3.83 a) displays one variation where three forms are combined at the dimensional center of the spirolateral. One is positioned horizontally and the other two are placed vertically. (Fig. 3.83 b) also combines three forms in the same manner, but at the beginning of the first turn of the spirolaterals. Another variation is to construct a volumetric interpretation, by combining six copies, each copy forms one side of the volume. In (Fig. 3.84 a) all the spirolaterals have the same orientation; an additional variation (Fig. 3.84 b), mirrored opposite sides, so that the spirolateral meets with common turns at the corners. This simple assembly concept attempted to investigate on how to translate the spirolateral flatness to a volumetric form.



a b

Fig. 3.83: Simple assembly using three copies.

[1] Ibid.

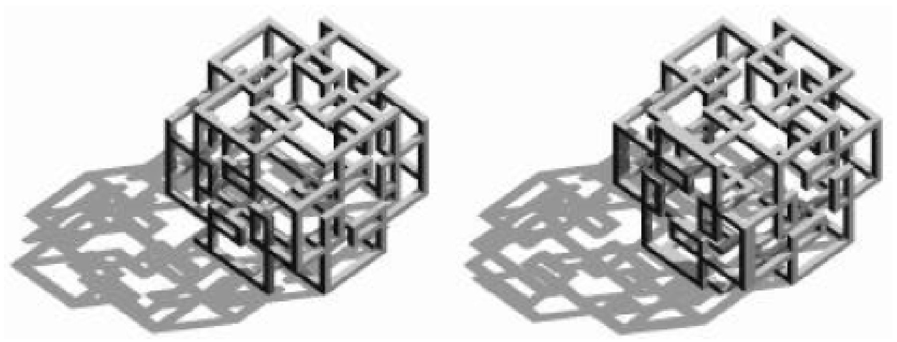

a b

Fig. 3.84: Simple assembly using six copies.

- **As a Construction**[1]

A set of construction can be generated by introducing vertical supports under the entire spirolateral, sets of turns, or each individual turn. These supports can have a height based on the turns or sets of turns. The first in the series considers the entire spirolateral. (Fig. 3.85 a) includes the spirolateral with supports at each end of each turn with an additional copy of it as a base support. (Fig. 3.85 b) follows the same concept except the supports are only at the beginning of each set of turns.

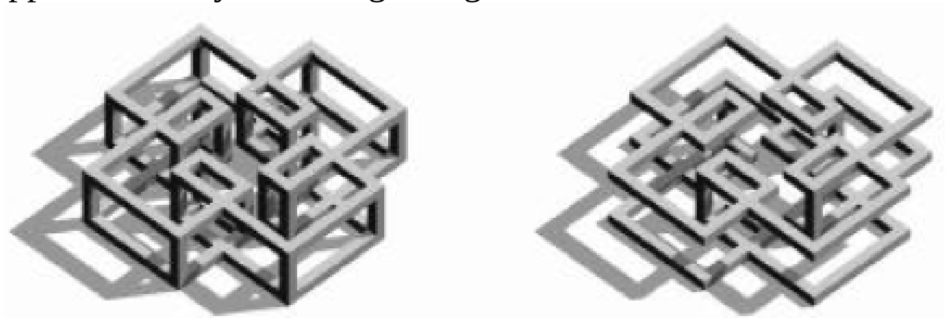

a b

Fig. 3.85: Simple construction using the entire spirolateral.

Following the concepts developed in the step and ramp forms, individual turns or a set of turns, the height of each turn is considered the first of these was individual turns, (Fig. 3.86 a) increases the height of each turn, segment by segment throughout the spirolateral. Supports are included at each start of each turn. (Fig. 3.86 b) is based on the same concept except that the height is decreased at the midpoint. (Fig. 3.86 c) also exhibits increasing

[1] Ibid.

and decreasing but the direction is changed at every set of turns. While (Fig. 3.87) shows excluding supports at some set of turns.

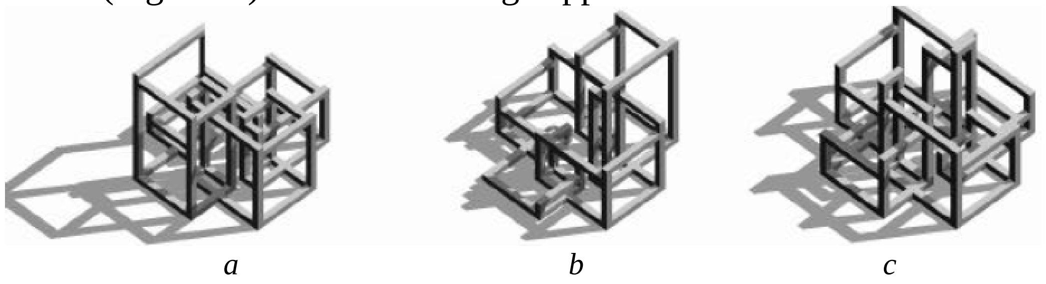


Fig. 3.86: Increasing/decreasing turn height.

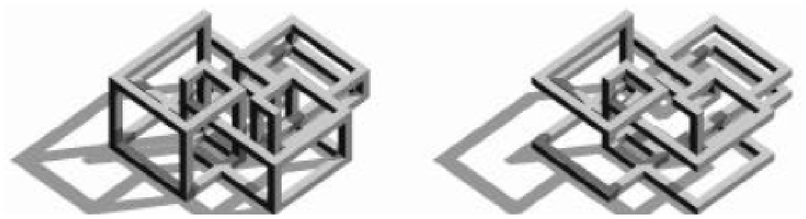


Fig. 3.87: Excluding supports at some set of turns.

- **Using Spirospaces entity**[1]

A Spirospace is a tridimentional geometrical entity characterized by its formal configuration. It is inspired from the bidimentional geometries of spirolaterals. The main value of Spirospaces for architectural design resides on its spatial potentiality. It is easy to verify from a simple observation of a spirospace (Fig. 3.88, 3.89), its strong analogy with architectural forms.

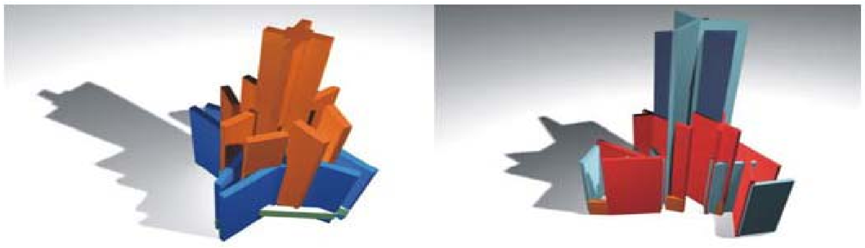


Fig. 3.88: Typical Spirospaces.

[1] Luis, F.B., Roberto, G.L. and Roberto, S.: *Spirospaces in Architectural Design.* .Design Systems Laboratory, University of Tucuman ,Argentina, 1st ASCAAD International Conference, e-Design in Architecture KFUPM, Dhahran, Saudi Arabia. December 2004.

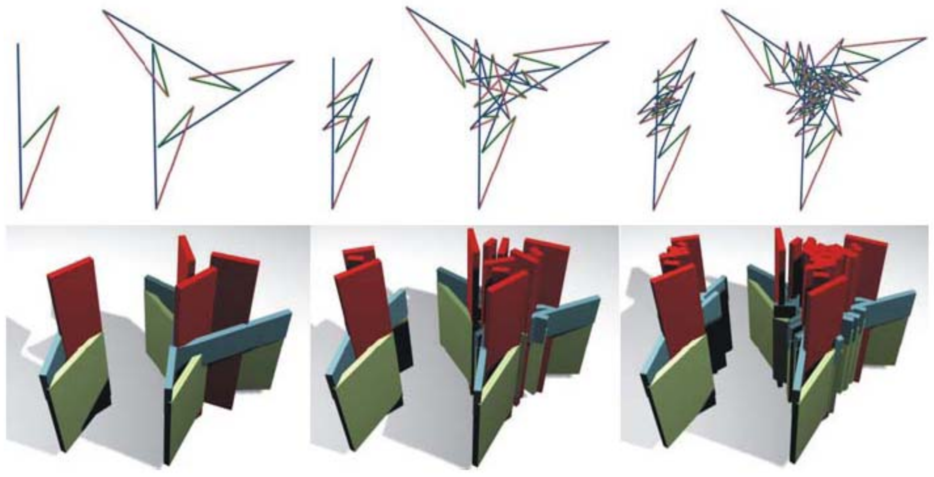


Fig. 3.89: Three first steps of a fractalized Spirospace.

- **Spirospaces Components**[1]

A spirospace is made up of tridimentional units, grouped by "packages" of variable complexity. A first package is compound by a set of "Pieces", somehow equivalent to spirolaterals segments. A second package is the "Module", conformed by consecutive pieces of variable length. A third package defines an "Element", which groups modules together (Fig. 3.90). The integration of elements can continue indefinitely, generating even more complex elements.

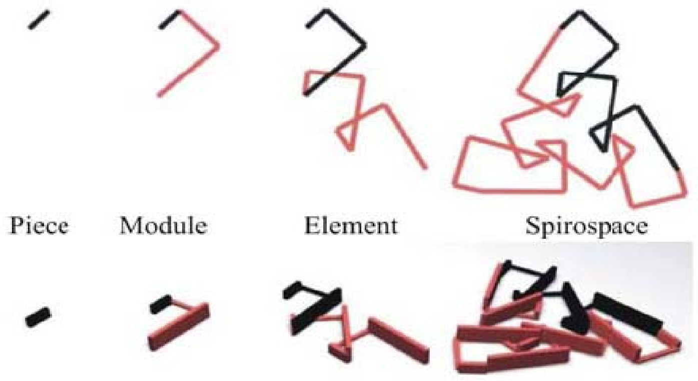


Fig. 3.90: Spirospaces components.

[1] Ibid.

- **Spirospaces Parameters**[1]

Spirospaces parameters are directly related with the components that define a spirospace, which are: pieces, joints and spaces. Parameters that operate on the "pieces" are:

(a) Shape Parameter: defines the geometry of spirospaces' pieces (Fig. 3.91).

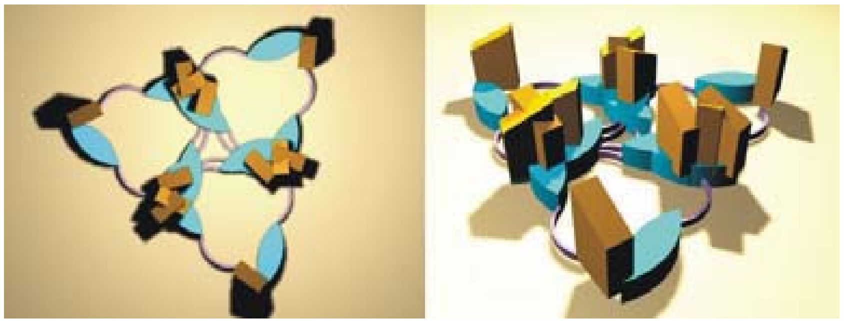


Fig. 3.91: Spirospace shape parameter.

(b) Dimension Parameter: defines the size of spirospaces' pieces (Fig. 3.92).

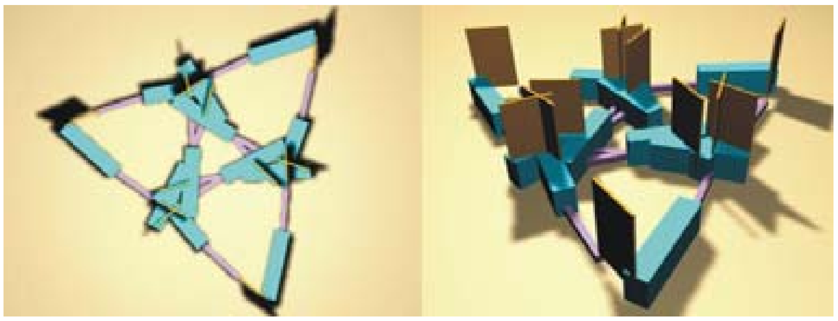


Fig. 3.92: Spirospace dimension parameter.

(c) Orientation Parameter: controls the rotation angle for each piece that composes a spirospace. The angle variation is only possible along the longitudinal axis of each piece (Fig. 3.93).

[1] Ibid.

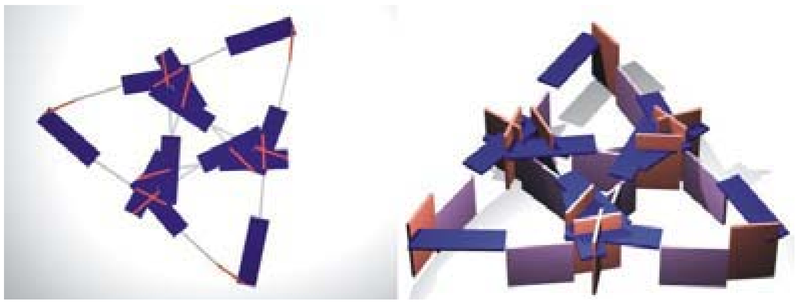


Fig. 3.93: Spirospace orientation parameter.

(d) Materiality Parameter: refers to the appearance of spirospaces' pieces (Fig. 3.94).

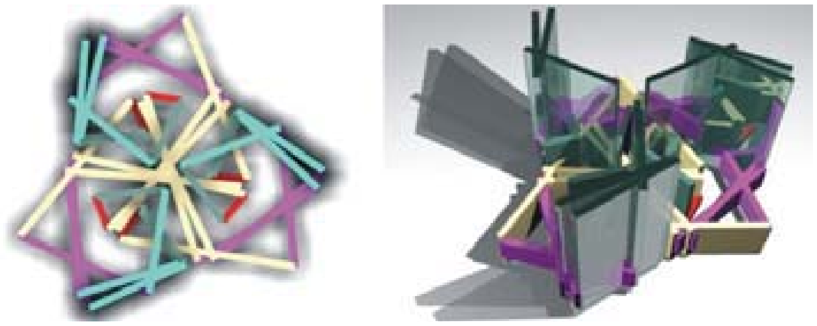


Fig. 3.94: Spirospace materiality parameter.

(e) Joints Parameter: joints parameters are based on the relative positions of concurrent pieces. Considering plan and elevation relative positions, it is proposed the distinction between "Meetings" (plan), and "Encounters" (elevations).

"Meetings" parameter expresses all possible combinations of joints between two consecutive pieces. (Fig. 3.95). exemplifies this.

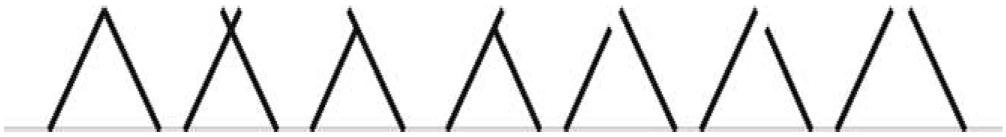


Fig. 3.95: Spirospace meetings types.

Spirospace spatiality varies with the type of junction. (Fig. 3.96) shows the same spirospace with different junctions.

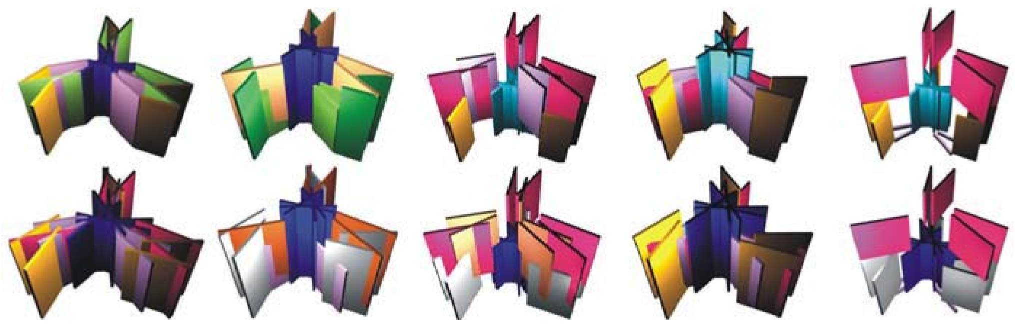


Fig. 3.96: Spirospace junctions.

The analogy process implies to relate two organizations, and to translate some characteristics from one to the other. Spirospaces proposed as the analogy source, and the design idea as the target. Now the question is, what to get from spirospaces that can be translated into architectural ideas?

- **The Analogy Source**[1]

A spirospace can be a very intricate entity, it is always perceived with a strong sense of unity, provided by its generation rules. For this reason, postulate that the characteristics related to the wholeness sense of spirospaces are valuable on three basic features: geometry, space, and syntax organization.

(a) Geometry: it refers to the morphological characteristics of a spirospace. The spirospaces arrangements usually have a remarkable formal personality, obtained from their generation rules, which means that this aspect has a powerful analogy potentiality.

[1] Ibid.

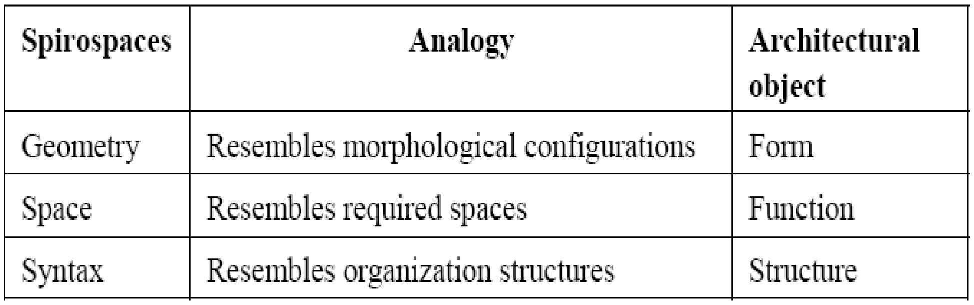
(b) Syntax: it refers to the conceptual organization of a spirospace. It implies a high degree of abstraction from direct observation. It is the form structure of the spirospace.

(c) Space: it refers to the space generated by spirospaces. It implies the observer interpretation to consider either objectual or interstitial spaces configurations, mentioned before.

- **The Analogy Target**[1]

Starting from a conceptual definition it is possible to associate some features of the object used as a creative trigger, establishing a relation by analogy. Creative analogy is accomplished by translating some characteristic from the paradigmatic entity used as a creative trigger to some characteristic of the architectural ideas. The form, function and structure are the target for the creative analogy. This means that the geometry of a spirospace can be used to inspire architectural morphological configurations.

We can use spirospaces to interpret some functional architectural schemes; those related to a central system with rotational subsystems, or to use the syntax organization of a spirospace to inspire the layout of the same kind of architectural ideas. The following table expresses all possible analogies between a spirospace and an architectural object (Tab. 3.1).



Spirospaces	Analogy	Architectural object
Geometry	Resembles morphological configurations	Form
Space	Resembles required spaces	Function
Syntax	Resembles organization structures	Structure

Tab. 3.1: A table showing possible analogies between a spirospace and an architectural object.

[1] Ibid.

Summary of chapter three:

Chapter three discussed two generative tools with clearly different outcomes belonging to a new branch of mathematics known by Chaos mathematics, and opposed in its conception to the conventional mathematical and geometrical principles preponderant until this moment. In this method of generation, forms do not born in the designer's mind but they are generated using iterative random functions and recursive equations in a computational medium. The designers only choose, select, evaluate and transform these forms. Fractals and Spirolaterals are the two studied examples of these chaotic phenomena. The chapter Studied: their nature, mathematical concepts and architectural potentials, with the purpose to use them as form generators in the preliminary phase of the form finding process.

Chapter 4
Evolutionary Based Mathematical Generative Systems

Chapter 4: Evolutionary Based Mathematical Generative Systems

Since Charles Darwin proposed his theory stating that all species are generated via the process of evolution, Attempts had been developed to explain the adaptive process of natural systems and to design artificial systems based upon these natural ones[1] (Fig.4.1).

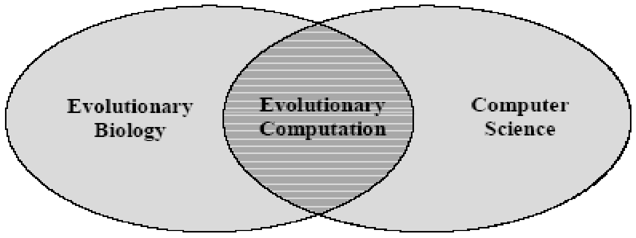


Fig. 4.1: Evolutionary computation roots in computer science and Evolutionary Biology.

Evolutionary design has its roots in computer science, design, and evolutionary biology. It is a branch of evolutionary computation, which extends and combines CAD and analysis software, and borrows ideas from natural evolution[2] (Fig. 4.2).

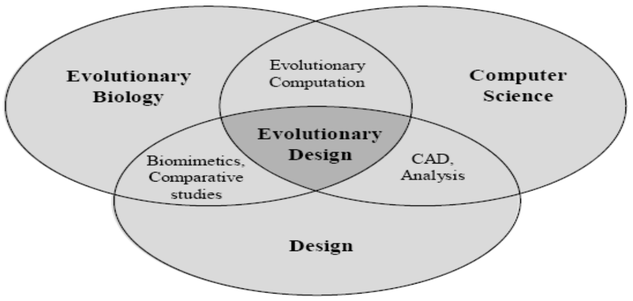


Fig. 4.2: Evolutionary design roots in Computer Science, Design and Evolutionary Biology.

Evolutionary design covers generative computer tools useful in the simulation and generation of 3D graphics of higher level of complexity and diversity. Examples of these evolutionary tools are: Genetic algorithms, Cellular automata.

[1] Bentley, P: *Evolutionary Design by Computers*. Morgan Kaufmann publishers, 1999.
[2] Eldaly, H.: *Architecture in the Age of Information Technology*. M.Sc. thesis in architecture, Ain Shams University, 2005.

4-1 Genetic Algorithms

4-1-1 Introduction

4-1-1-1 Definition

Genetic algorithm is an artificial intelligence procedure based on the principles of evolution and natural selection. i.e., survival of the fittest. GA is inspired by natural systems and how they often rely on the repetition of very simple steps such as crystal growth.

It is now considered not only powerful enough to solve biological puzzles, but also tools useful in the simulation of algorithms and 3D graphics having a huge potential for creating artistic forms of higher level of complexity and diversity. GA today is the most widely used form of the evolutionary algorithm.

4-1-1-2 Outlines

GA originally developed and characterized by John Holland[1] (1975) and made famous by David Goldberg[2].

The first experimental system for art generation, 'FormSynth' was created by William Latham in (1989). It was a system for drawing on paper in which repeated applications of rules generate an evolutionary tree of unexpected forms[3], some of

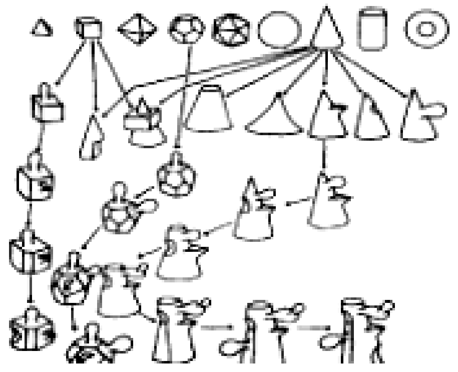


Fig. 4.3: Small hand-drawn genetic tree.

[1] Holland, J. H.: *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
[2] Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, 1989.
[3] Jakimowicz, A., Barrallo, J. and Guedes, E.M: *Spatial Computer Abstraction: From Intuition to Genetic Algorithms*. CAAD futures Digital Proceedings, 1997.

them with architectural quality. (Fig .4.3) presents an image based on 'Mutator', an algorithm created by William Latham and Stephen Todd[1]. This algorithm displays geometric forms under the action of associated gene values.

The basis of the graphic process lies in a very simple gallery of Euclidean objects (cubes, cylinders, pyramids, spheres, ...) and a collection of mutation processes based on natural systems (tree branch, spider web, DNA helix, ...).

The combined action of one or more of these objects with a mutation process results in new forms with an imaginative and increasingly complex structure. The user must explore the resulting forms and select one of them, (The user acts as a judge driving selection, using aesthetic judgments to breed artwork). So only the most aesthetic form is allowed to survive and reproduce.

GAs are widely used in optimization of design in many engineering fields, to improve a previous design, or to create new design from scratch. GAs show great power in design fields due to its ability of creating a wide range of alternatives in design, in a very short time, which can help the designer in decision making.

The idea of the GAs is based mainly on the genetic rules, similar to the genetic rules of the living creatures.

4-1-1-3 The Evolutionary Concept

Genetic algorithms (GAs) are algorithms inspired from Darwin's theory of evolution. These algorithms operate with the population of candidate solutions (individuals). Every new population is formed using genetically inspired operators (like crossover and mutation) and through a selection pressure, which guides the evolution towards better areas of the search space. The evolutionary algorithms receive this guidance by evaluating every

[1] Todd, S. and Latham, W.: Evolutionary Art and Computers. Academic Press, 1992.

candidate solution to define its fitness. The fitness, calculated by the fitness function (i.e. objective function), indicates how well the solution fulfills the problem objective (specification).

Genetic algorithms use two separate spaces: the search space and the solution space[1] (Fig. 4.4).

i. The search space is space of coded solutions to the problem.

ii. The solution space is the space of actual solutions.

Coded solutions, or genotypes must be mapped as actual solutions, or phenotypes, before the quality or fitness of each solution can be evaluated.

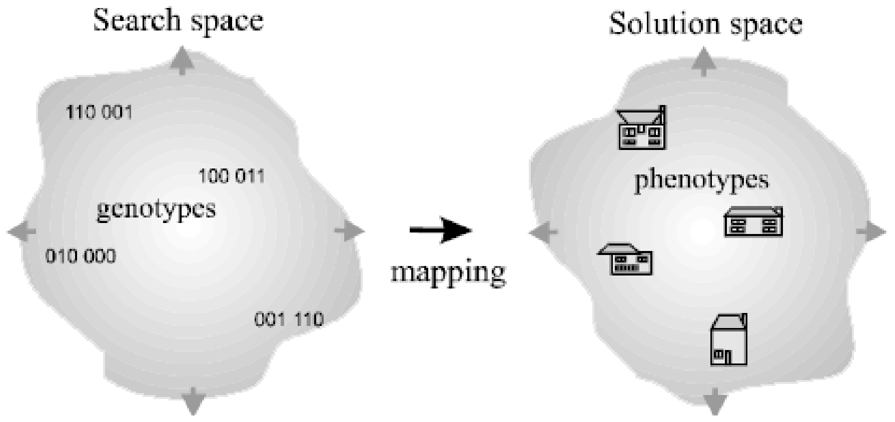


Fig. 4.4: Mapping genotypes in the search space to phenotypes in the solution space.

In GA each individual element (in the phenotype) in the solution space, takes a code (a binary code depends on its properties) in the search space (in the genotypes). Phenotypes usually consist of collections of parameters; Genotypes consist of coded versions of these parameters. A coded parameter is normally referred as a gene, with the values a gene can be known as alleles. A collection of genes in one genotype is often held internally as a string, and is known as a chromosome (Fig. 4.5, 4.6).

[1] Bentley. P.: *Evolutionary Design by Computers*. Morgan Kaufmann publishers, 1999. op.cit.

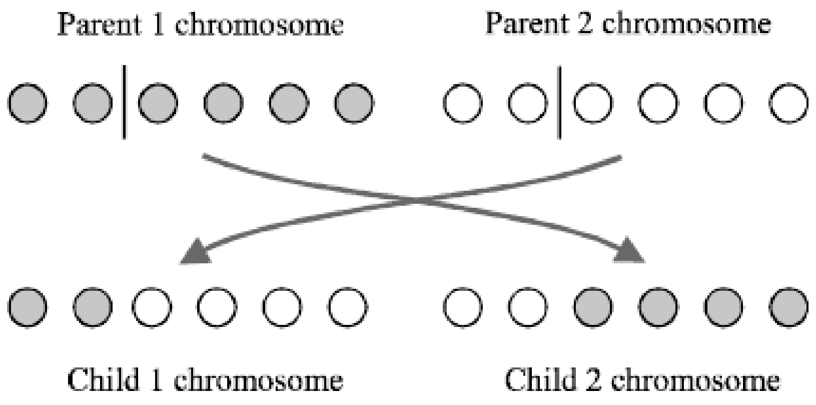


Fig. 4.5: The behavior of the crossover operator. The vertical line shows the position of the random crossover point.

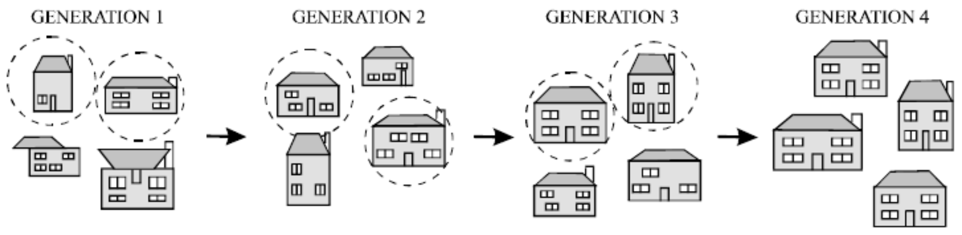


Fig. 4.6: Four generations of evolving house designs using a population size of four. Parents of the next generation are circled.

The simplest form of a GA is summarized in (Fig. 4.7). This genetic algorithm can be explained in the following points[1]:

i. The genotype of every individual in the population is initialized with random alleles.

ii. The main loop of the algorithm then begins, with the corresponding phenotype of every individual in the population being evaluated and given a fitness value according to how well it fulfils the problem objective or fitness function.

iii. These scores are then used to determine how many copies of each individual are placed into a temporary area often

[1] Ibid.

termed the 'mating pool' (i.e. the higher the fitness, the more copies that are made of an individual).

iv. Two parents are then randomly picked from this area.

v. Offspring are generated by the use of the crossover operator, which randomly allocates genes from each parent's genotype to each offspring's genotype. For example, given two parents: 'ABCDEF' and 'abcdef', can create a new generation of 'ABcdef' and 'abCDEF', and another new generation can be created by 'ABcdef' and 'abCDEF', and so on…..

vi. This entire process of evaluation and reproduction then continues until, either a satisfactory solution emerges or the GA will run for more generations.

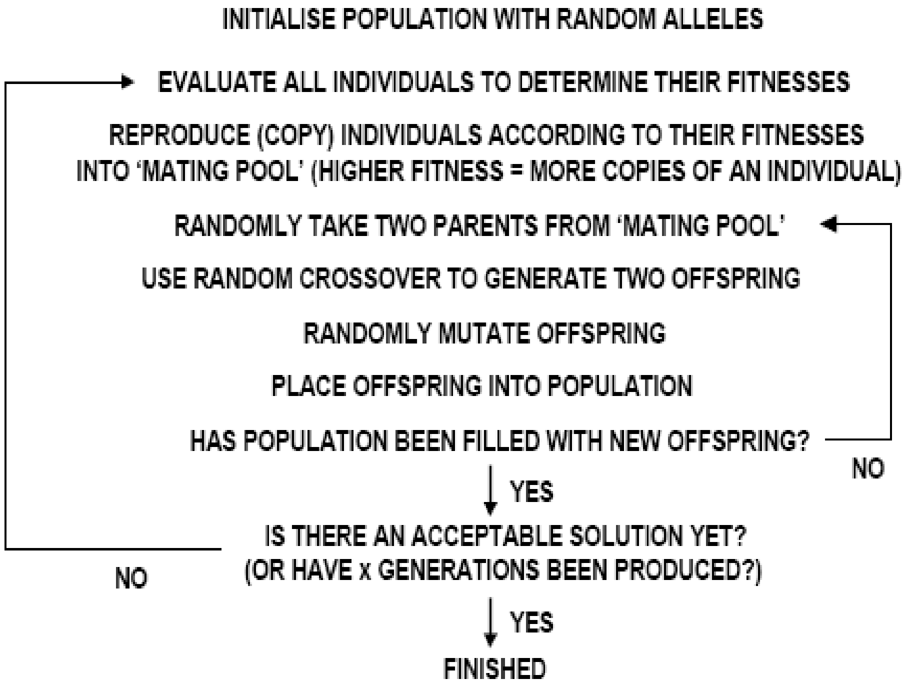

INITIALISE POPULATION WITH RANDOM ALLELES
EVALUATE ALL INDIVIDUALS TO DETERMINE THEIR FITNESSES
REPRODUCE (COPY) INDIVIDUALS ACCORDING TO THEIR FITNESSES INTO 'MATING POOL' (HIGHER FITNESS = MORE COPIES OF AN INDIVIDUAL)
RANDOMLY TAKE TWO PARENTS FROM 'MATING POOL'
USE RANDOM CROSSOVER TO GENERATE TWO OFFSPRING
RANDOMLY MUTATE OFFSPRING
PLACE OFFSPRING INTO POPULATION
HAS POPULATION BEEN FILLED WITH NEW OFFSPRING?
NO
YES
IS THERE AN ACCEPTABLE SOLUTION YET? (OR HAVE x GENERATIONS BEEN PRODUCED?)
NO
YES
FINISHED

Fig. 4.7: The simple genetic algorithm[1].

[1] Eldaly. H.: *Architecture in the Age of Information Technology*. M.Sc. thesis in architecture, Ain Shams University, 2005. op.cit.

4-1-2 Classifications

The use of evolutionary computation by (GAs) to generate designs has taken place in many different fields over the last 10 or 15 years.

Designers have optimized selected parts of their designs using evolution, artists have used evolution to generate aesthetically pleasing forms, architects have evolved new building plans from scratch, and computer scientists have evolved morphologies and control systems of artificial life[1].

In general, these types of evolutionary design by (GAs) can be classified into many categories the most important are as follows: Evolutionary design optimization, Evolutionary art, Evolutionary artificial life forms and Creative evolutionary design (Fig. 4.8).

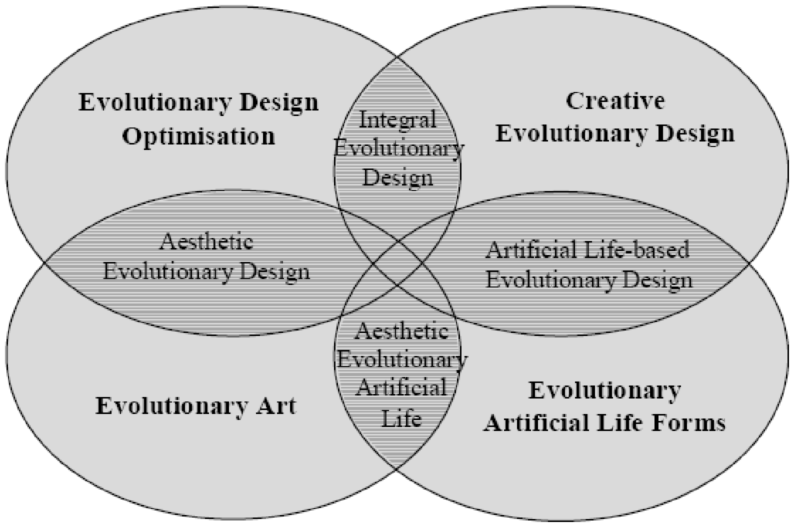
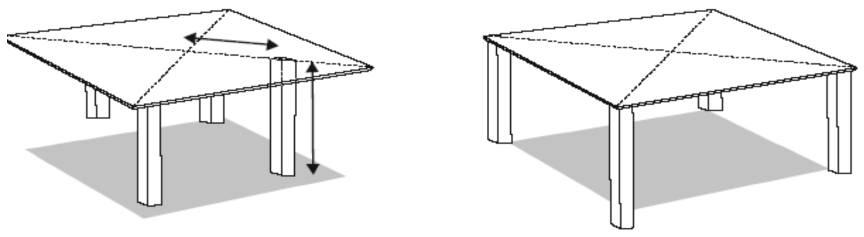

Evolutionary Design Optimisation
Integral Evolutionary Design
Creative Evolutionary Design
Aesthetic Evolutionary Design
Artificial Life-based Evolutionary Design
Aesthetic Evolutionary Artificial Life
Evolutionary Art
Evolutionary Artificial Life Forms

Fig. 4.8: Classifications of evolutionary design by GAs.

1 Bentley. P.: *Evolutionary Design by Computers*. Morgan Kaufmann publishers, 1999. op.cit.

4-1-2-1 Evolutionary Design Optimization

For example, Evolutionary optimization of a table (Fig. 4.9).



Phenotype:
Table consisting of fixed top and four legs defined by:
Length of leg 1, Distance of leg 1 from centre
Length of leg 2, Distance of leg 2 from centre
Length of leg 3, Distance of leg 3 from centre
Length of leg 4, Distance of leg 4 from centre

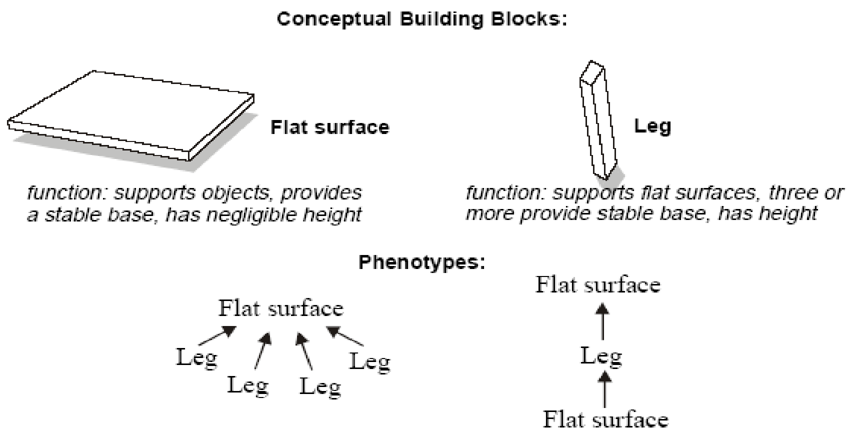
Genotype:

11010110	10101101	10101110	10011010	01101010	10001010	11110010	00101110
Length 1	*Distance 1*	*Length 2*	*Distance 2*	*Length 3*	*Distance 3*	*Length 4*	*Distance 4*

Fig. 4.9: Evolutionary optimization of a table.

4-1-2-2 Conceptual Evolutionary Design

For example, Conceptual evolutionary design of table (Fig. 4.10).


Conceptual Building Blocks:
Flat surface
Leg
function: supports objects, provides a stable base, has negligible height
function: supports flat surfaces, three or more provide stable base, has height
Phenotypes:
Flat surface
Leg
Leg
Leg
Leg
Flat surface
Leg
Flat surface

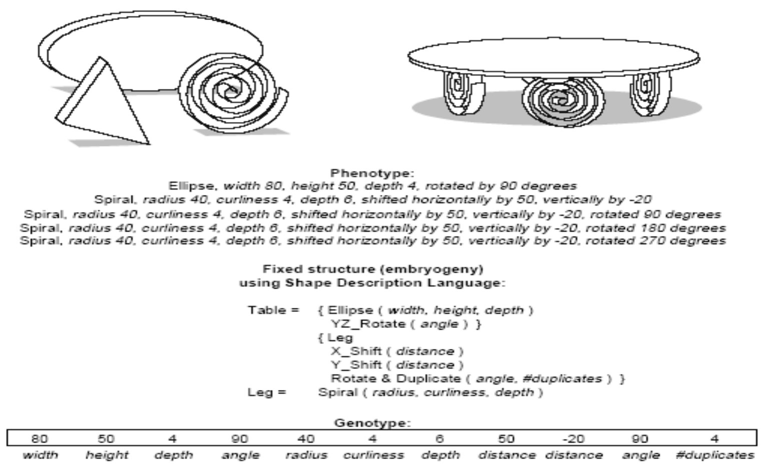
Genotype:

0000 1111	0001 0000	0001 0000	0001 0000	0001 0000
Concept1 Ptr1	*Concept2 Ptr2*	*Concept3 Ptr3*	*Concept4 Ptr4*	*Concept5 Ptr5*

Fig. 4.10: Conceptual evolutionary design of a table.

4-1-2-3 Evolutionary Art

For example, evolving artistic tables (Fig. 4.11).



Phenotype:
Ellipse, *width 80, height 50, depth 4, rotated by 90 degrees*
Spiral, *radius 40, curliness 4, depth 6, shifted horizontally by 50, vertically by -20*
Spiral, *radius 40, curliness 4, depth 6, shifted horizontally by 50, vertically by -20, rotated 90 degrees*
Spiral, *radius 40, curliness 4, depth 6, shifted horizontally by 50, vertically by -20, rotated 180 degrees*
Spiral, *radius 40, curliness 4, depth 6, shifted horizontally by 50, vertically by -20, rotated 270 degrees*

Fixed structure (embryogeny)
using Shape Description Language:

```
Table =   { Ellipse ( width, height, depth )
              YZ_Rotate ( angle ) }
          { Leg
              X_Shift ( distance )
              Y_Shift ( distance )
              Rotate & Duplicate ( angle, #duplicates ) }
Leg =     Spiral ( radius, curliness, depth )
```

Genotype:

80	50	4	90	40	4	6	50	-20	90	4
width	*height*	*depth*	*angle*	*radius*	*curliness*	*depth*	*distance*	*distance*	*angle*	*#duplicates*

Fig. 4.11: Evolving artistic tables.

Evolving artistic chairs using artificial DNA by Soddu[1] (Fig. 4.12).

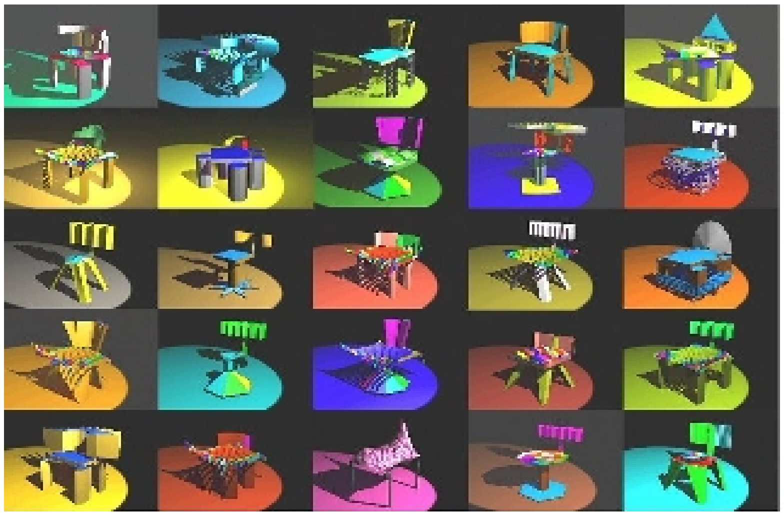


Fig. 4.12: Generated artistic chairs with "artificial DNA" by Soddu[2].

[1] Celestino Soddu, A professor of Architectural Design at Politecnico di Milano University, Italy, Director of Generative Design Lab, DiAP and Chairman of Generative Art Annual International Conference

[2] Soddu, C.: *Generative Natural Flux*. Proceedings of Generative Art Conference, Milan, AleaDesign Publisher, December 2001.

Celestino Soddu, an Italian architect and chairman of the generative art annual international conference held in Milan developed many AI software for the purpose of generating endless variations of artificial DNA species in art and architecture.

Another example for an evolutionary art application (Fig. 4.13) shows variations for a woman portrait using genetic interpretative code that characterizes Picasso's paintings made by Soddu.

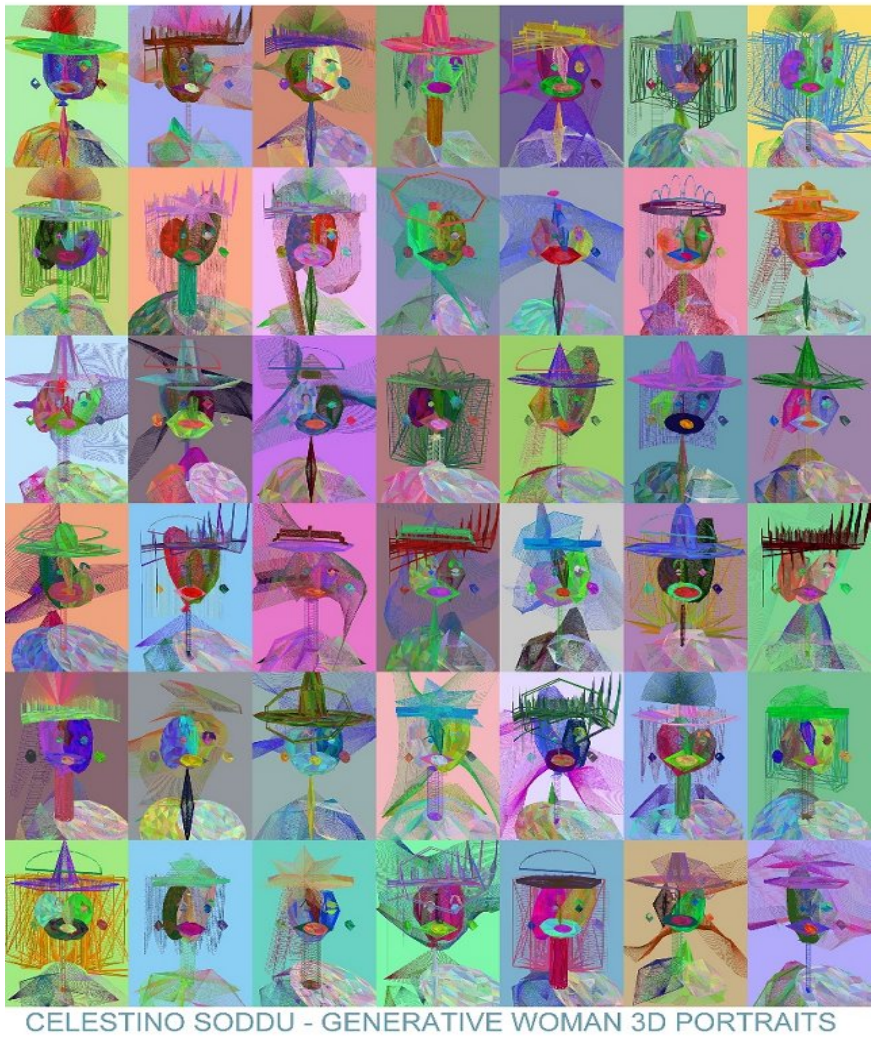


Fig. 4.13: Generated portraits “d’après Picasso” with “artificial DNA” by Soddu[1].

[1] http://www.generativeart.com. Accessed 15/7/2007.

Soddu also used GA to generate species of lamps (Fig. 4.14), coffee pots, rings (Fig. 4.15), chairs and many other artistic products.

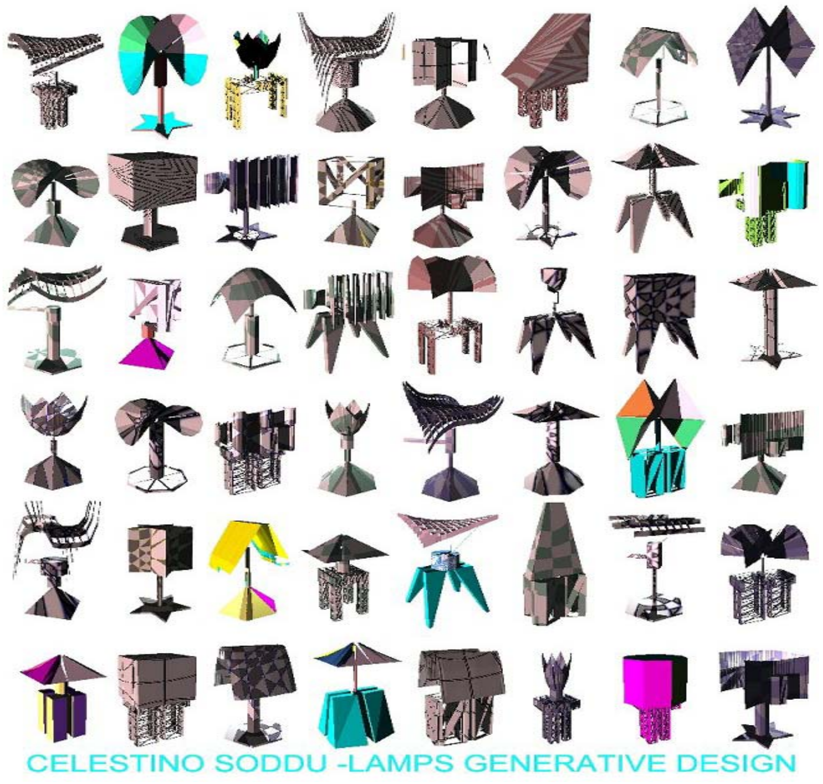


Fig. 4.14: Generated artistic lamps with “artificial DNA” by Soddu[1].

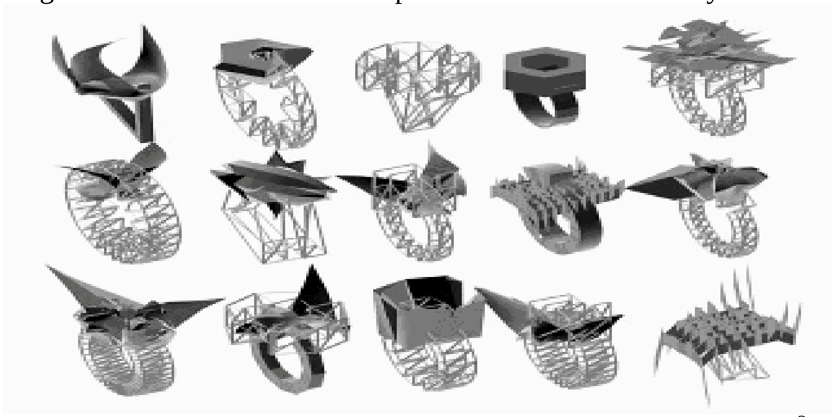


Fig. 4.15: Generated artistic rings with “artificial DNA” by Soddu[2].

[1] Soddu, C.: *Generative Natural Flux*. Proceedings of Generative Art Conference, Milan, AleaDesign Publisher, December 2001. op.cit.
[2] Ibid.

4-1-2-4 Evolutionary Artificial Life Forms

GAs spread widely in industrial design but so far slowly in architecture design. One of the famous uses in industry is furniture design. The design criteria have been translated into genotypes and the products are left to an autonomous process. After generating sufficient alternatives, form selection can be made according to the user needs, material criteria or an expert decision. Structural and functional details will be elaborated after the selection.

Generative evolutionary design of a table (Fig. 4.16) and generated chairs using 'ConGen' software (Fig. 4.17).

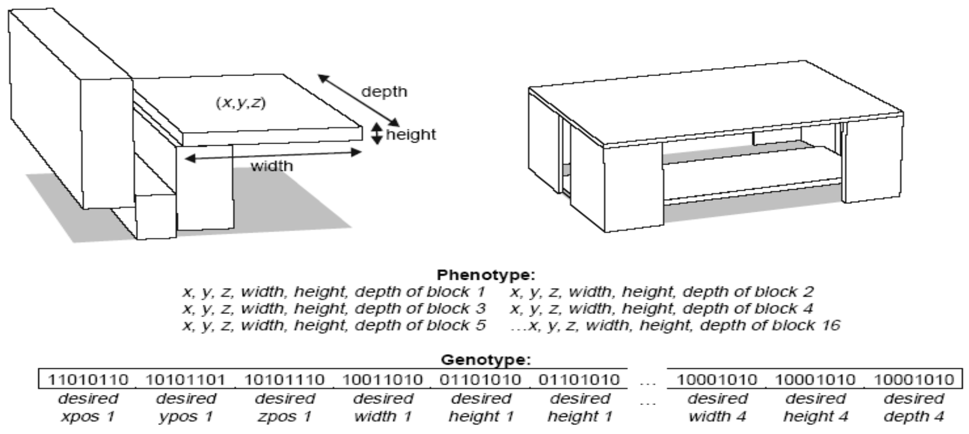

(x,y,z)
depth
height
width
Phenotype:
x, y, z, width, height, depth of block 1   x, y, z, width, height, depth of block 2
x, y, z, width, height, depth of block 3   x, y, z, width, height, depth of block 4
x, y, z, width, height, depth of block 5   ...x, y, z, width, height, depth of block 16
Genotype:
11010110 | 10101101 | 10101110 | 10011010 | 01101010 | 01101010 | ... | 10001010 | 10001010 | 10001010
desired xpos 1 | desired ypos 1 | desired zpos 1 | desired width 1 | desired height 1 | desired height 1 | ... | desired width 4 | desired height 4 | desired depth 4

Fig. 4.16: Generative evolutionary design of a table.

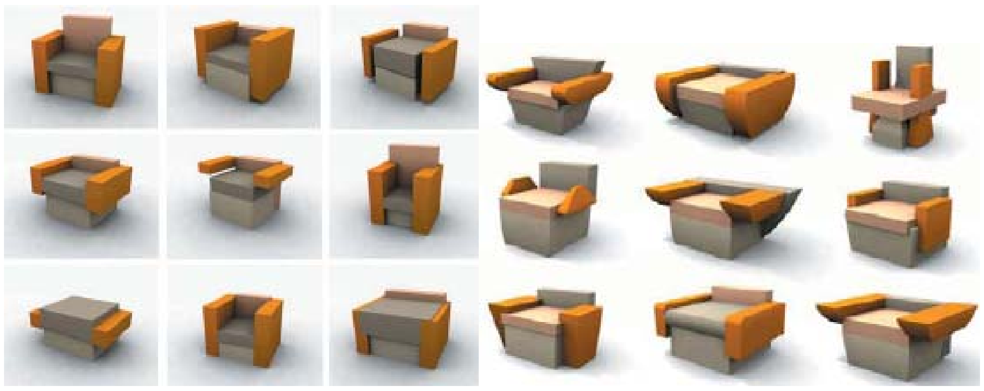


Fig. 4.17: Generated chairs using 'CongGen' software.

4-1-3 Architectural Potentials

The previous types of GA applications (evolutionary design optimization, creative evolutionary design, etc.), can be applied in simple form finding problems, but functional problems still require more complicated computing power.

GA proposed the evolutionary model of nature as the generating process for architectural form. The creative power of natural evolution is done by generating virtual architectural models. Architecture by a GA is considered as a form of artificial life.

The use of genetic algorithms to manufacture forms and relationships is the main process in creating an evolutionary architecture. These genetic algorithms can be used to generate complex spatial models, which can then be filled, punched, and stretched to meet other functional or aesthetic criteria.

- **Architectural concepts in evolutionary architecture are expressed as[1]:**

i. Generative rules, so that their evolution and development can be accelerated and tested by the use of computer models. Computer models are used to create the development of prototypical forms that are then evaluated on the basis of their performance in a simulated environment. The best models become (according to their performance) the parents, which are going to create better models in the new offspring. These new evolutionary steps (offspring) can be generated in a short space of time and the emergent forms are often unexpected.

ii. Genetic language that produces a code script (Genotype at search space) of instructions for form-generation.

[1] Frazer, J., Frazer, J., Liu, XY., Tang, MX. and Janssen, P.: *Generative and Evolutionary Techniques for Building Envelope Design*. GA2002 (Generative Art and Design Conference, Politecnico di Milano University, Italy , Milan 11-12-13 December 2002).

- **Applying GA to design problems**

Applying GA to design problems means treating design variants as members of a population of candidate solutions that compete for survival in a game of evolution. Evolution proceeds in two steps: reproduction and selection. Reproduction means members of the population mate and generate offspring. Selection, in the context of genetics, means that the individual phenotypes undergo a process of testing their 'fitness'. In biology, fitness is the capability of the individual to survive and reproduce[1].

- **How do we make design variants mate?**

By treating the coding of their parts as genes capable of being taken apart and recombined in certain ways. Mating is accomplished by crossing over the strings of bits encoding two 'parent' design (Fig. 4.18). The reproduction process engendered by GA is 'blind' in that both the selection of the pairs that mate and the selection of the locations of the strings of bits at which the crossing over takes place are random[2].

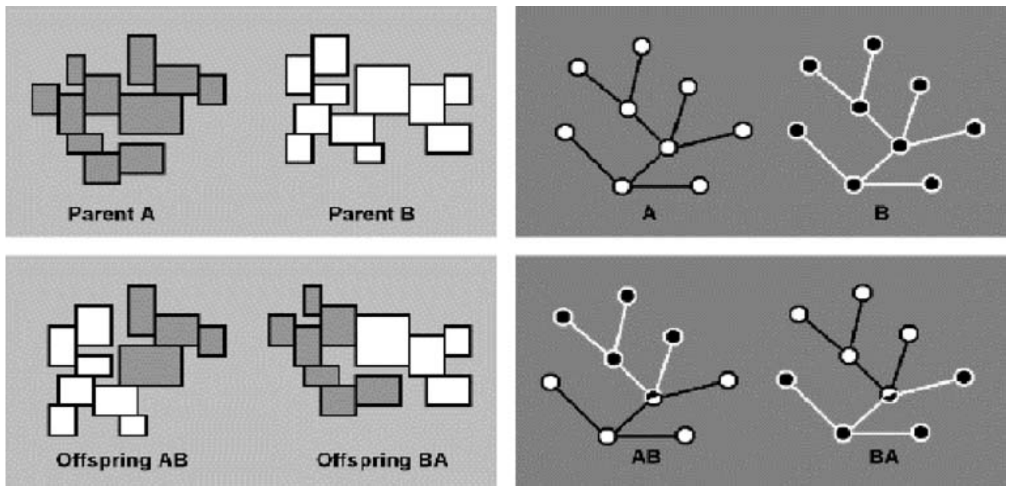


Fig. 4.18: Crossing over design variants.

[1] Elezkurtaj, T. and Franck, G.: *Genetic Algorithms in Support of Creative Architectural Design*. Vienna University of Technology, Department of Computer Aided Planning and Architecture, Vienna, Austria, eCAADe conference proceeding, Liverpool (UK) 15-17 September 1999.
[2] Ibid.

In addition to this two kind of randomization, the strings are subject to a third kind of random change, acting in the manner mutation does in genetics. In the context of GA, mutation means that single bits are changed by chance from 1 to 0 or vice versa.

4-1-3-1 Generating 2D Forms

The above concepts can be exemplified through the next example, showing how a designer can use GA in generating two dimensional forms. Generally, the evolutionary model requires that a design concept must be described in a genetic code. The code is then mutated and developed in a computer program into a series of models in response to a simulated environment. The models are then evaluated in the simulated environment and the code of successful models is selected. The selected code is then used to repeat the cycle until a particular stage of development is selected for prototyping in the real world.

- **A House space generation example**[1]

A house can be considered to be composed of a number of zones, such as living zone, entertainment zone, bed zone, utility zone, etc. Each zone is composed of a number of rooms (or spaces), such as living room, dining room, bedroom, hall, bathroom, etc. Each room is composed of a number of space units.

Generally, in a design such as a house, the space unit will be constant. The scale (level of abstraction) of the space unit depends on the precision required in differences between various possible room sizes. The smaller the unit, the longer the genotype for a given size of room but the greater the shape alternatives. But first some criteria must be described for a thorough understanding.

[1] Rosenman, M.A. and Gero, J.S.: *Evolving Designs by Generating Useful Complex Gene Structures*. In P. Bentley (ed.), Evolutionary Design by Computers, Morgan Kaufmann, London, pp. 345-364. 1999.

- **The Design Grammar**

In the above example, the generation of spaces, basically comes down to locate spatial component units for that level. At the room level, the component unit is a fundamental unit of space. At the zone level, the component unit is a room and at the house level the component unit is a zone[1].

The design grammar used here is based on the method for constructing polygonal shapes represented as closed loops of edge vectors. The grammar is based on a single fundamental rule which states that any two polygons, P_i and P_j, may be joined through the conjunction of negative edge vectors, V_1 and V_2, (equal in magnitude and opposite in direction). The conjoining of these vectors results in an internal edge and a new polygon, P_k. This rule ensures that new cells are always added at the perimeter of the new resultant shape.

The fundamental conjoining rule can be specialized for different types of geometries. Orthogonal geometries are based on the following four vectors of unit length: W = (1, 90), N = (1, 0), E = (1, 270), S= (1, 180) so that the two pairs of negative vectors are N - S and E - W. These two pairs of negative vectors allow for the generation of all polyminoes. Orthogonal geometries will be used in this example without loss of generality. Other (sub) rules may be formed for other geometries.

- **Genotype and Phenotype**

A polygon is described by its sequence of edge vectors. A suffix is used to identify individual edges of the same vector type. Thus the square cell is described as (W1, N1, E1, S1). The sequence of edge vectors for a shape is the phenotype providing the description of that shape's structure. The genotype for any generated polymino is the sequence of the two subshapes

[1] Ibid.

(polyminoes) used and the two edges joined. An example of the generation of a trimino is shown in[1] (Fig. 4.19).

(Fig. 4.14) shows a basic unit or cell, P1, which provides a starting point for the generation of polyminoes. Each generated shape is accompanied by its genotype and phenotype.

The generation of these polyminoes occurs from a random selection of edges in the first shape conjoined with a random selection from equal and opposite edges in the second shape. At each step in the generation, the phenotype is reinterpreted to generate a new edge vector description and the conjoining (sub) rules applied.

The genotype for the generated trimino is given as (P2, P1, N2|S1). This can be expanded as ((P1, P1, E1|W1), P1, N2|S1). When the same units are used for generation, the unit can be omitted and the genotype represented as the sequence of edge vector conjoining. That is P3(g) = (E1|W1, N2|S1). The length of the genotype depends on the size of the polymino to be generated, that is on the area of the polymino. This corresponds to required room sizes.

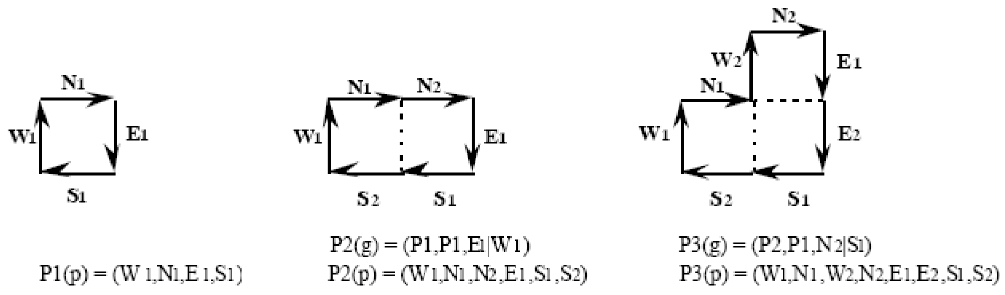


Fig. 4.19: Generation of a Trimino.

Once a population of different rooms is generated for each room type in a given zone, the zone can be generated through the conjoining of rooms in a progressive fashion. Because of the cell-type structure of the polygons, the conjoining may occur at any appropriate pair of cell edges. Therefore, a large number of

[1] Ibid.

possible zone forms can be generated from two rooms. An example of some possibilities arising from the conjoining of two polyminoes is given in (Fig. 4.20).

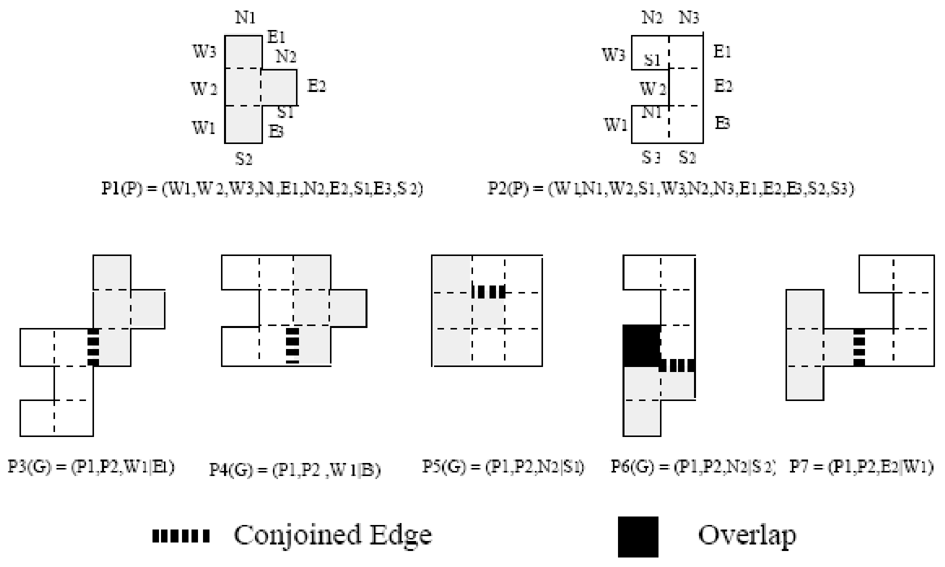


Fig. 4.20: Some Examples of Conjoining Two Polyminoes.

The two polyminoes, P1 and P2, represent instances of two different room types and the polyminoes resulting from the joining of the two rooms represent instances of a particular zone type. When one pair of edges are conjoined other edges may also be conjoined, e.g. P4, P5 and P6. In the case of overlap, as in P6, the resultant shape is discarded.

The same process used for generating zones is used to generate houses. The joining of different instances of different zone types generates different instances of houses.

The above grammar can be used to generate initial populations for each level in the spatial hierarchy. Each such initial population is then evolved, as necessary, so that solutions are 'adapted' to design requirements[1].

[1] Ibid.

• The Evaluation Criteria - Fitness Functions

At each level, different fitness functions apply according to the requirements for that level. While the requirements for designs of houses involve many factors, many of which cannot be quantified or adequately formulated in a fitness function, some simple factors have been used initially to test the feasibility of the approach. For this example, the fitness function for rooms consists of minimizing the perimeter to area ratio and the number of angles.

This requirement tends to produce compact forms, useful as rooms. For zones, the fitness function consists of minimizing a sum of adjacency requirements between rooms reflecting functional requirements.

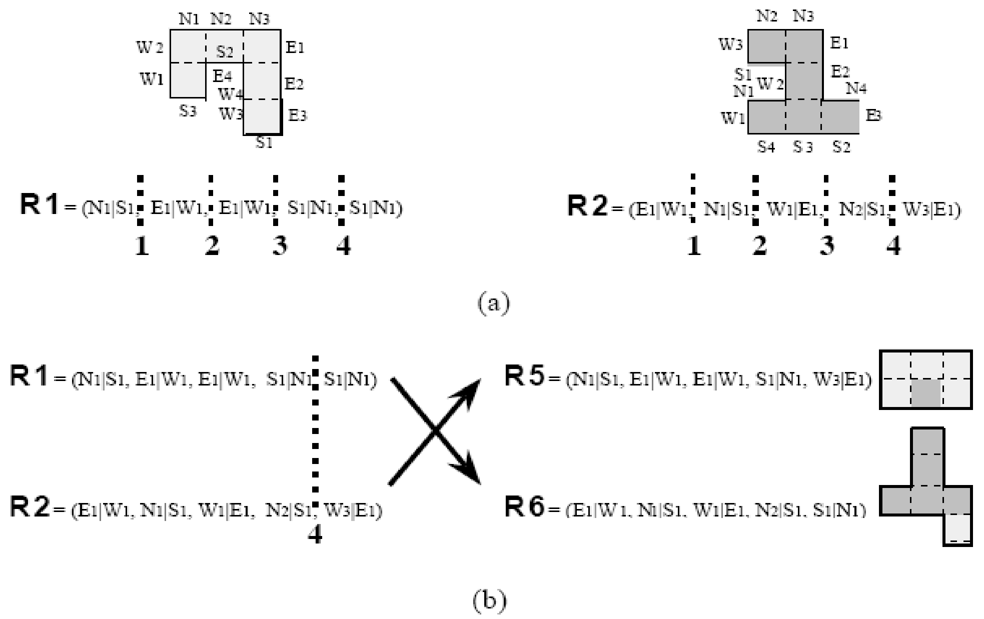
At the house level, the fitness function consists of minimizing a sum of adjacency requirements between rooms in one zone and rooms in other zones. This has the tendency to select those arrangements of zones where adjacency interrelations are required between rooms of different zones. In addition to these quantitative assessments, qualitative assessments will be made subjectively and interactively by a user/designer.

The aim is to direct the evolutionary process to produce populations of good solutions either as components for higher levels or at the final level itself. So that, even though the global optimum solution for the shape of a room using the above criteria, may be known, this may not be the optimum solution at the zone and house levels. By selecting other non-optimal but good solutions, according to the given criteria, good unexpected results may be achieved for the overall design[1].

[1] Ibid.

- **Propagation – Crossover**

Simple crossover is used for the production of 'child' members during the evolution process. Looking first at the room level to see the effect of such a crossover process, crossover can occur at any of the four sites as shown in (Fig. 4.21 a) with two results as shown in (Fig. 4.21 b). Since the cells are always of the same space unit, the cell identification in the genotype representation has been omitted for simplicity[1].



R1 = (N1|S1, E1|W1, E1|W1, S1|N1, S1|N1)
R2 = (E1|W1, N1|S1, W1|E1, N2|S1, W3|E1)
1 2 3 4
(a)
R5 = (N1|S1, E1|W1, E1|W1, S1|N1, W3|E1)
R6 = (E1|W1, N1|S1, W1|E1, N2|S1, S1|N1)
4
(b)

Fig. 4.21: Crossover at Room Level; (a) initial rooms R1 and R2 generated from unit square cell U1, (b) crossover at site 4.

At the zone level, crossover occurs as shown in (Fig. 4.22). Two initial instances of living zones, Z1 and Z2 are shown in (Fig. 4.22 a). Each zone has one instance of each of living room, dining room and entrance. (Fig. 4.22 B) shows crossover for one of the four possible sites. A similar process is followed at the house level.

[1] Ibid.

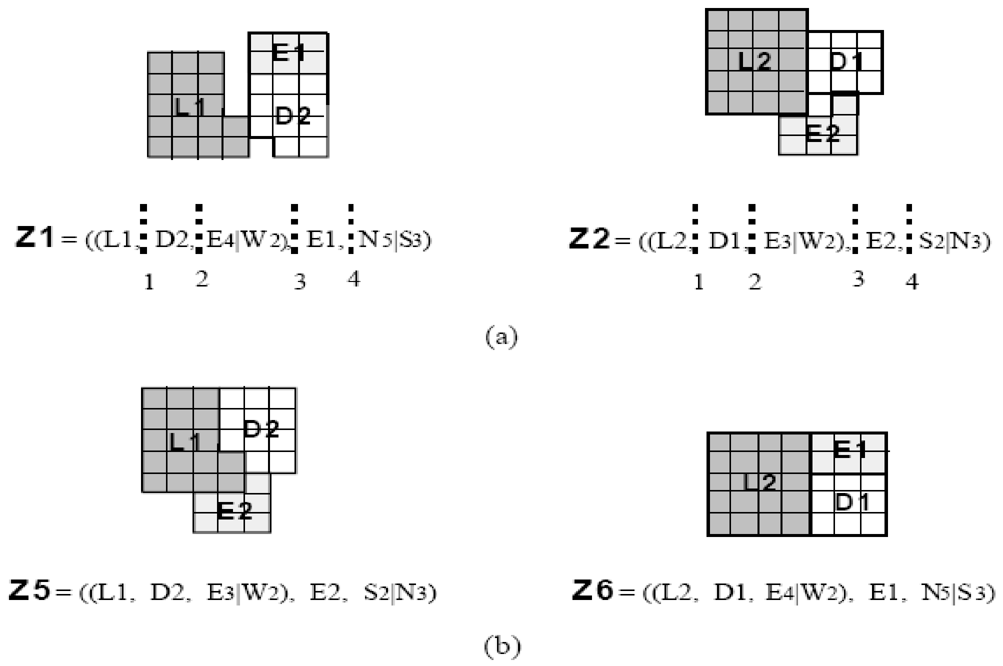

E1
L1
D2
L2
D1
E2
Z1 = ((L1, D2, E4|W2), E1, N5|S3)
1 2 3 4
Z2 = ((L2, D1, E3|W2), E2, S2|N3)
1 2 3 4
(a)
L1
D2
E2
L2
E1
D1
Z5 = ((L1, D2, E3|W2), E2, S2|N3)
Z6 = ((L2, D1, E4|W2), E1, N5|S3)
(b)

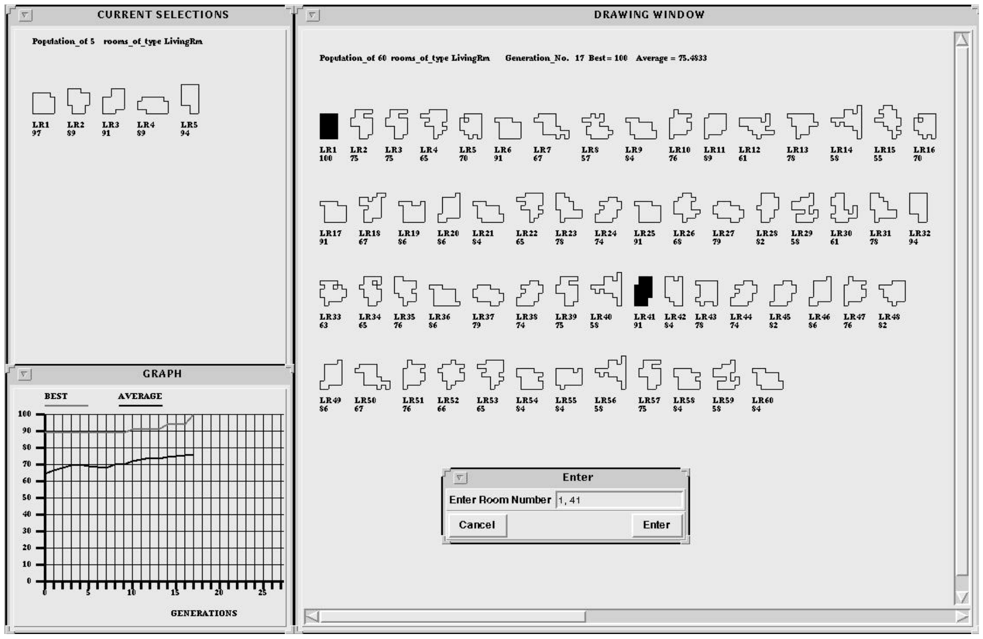
Fig. 4.22: Zone Crossover; (a) rooms and initial zones, Z1 and Z2, (b) crossover at Site 2.

- **Implementations**

A computer program written in C++ and Tcl-Tk under the Sun Solaris environment has been implemented using the simple criteria described previously. Each evolution run, for all levels, tends to converge fairly quickly to some dominant solution. Rather than use a mutation operator to break out of such convergence, it was found that a more efficient strategy was to generate multiple runs with different initial randomly generated populations. This produces a variety of gene pools thus covering a more diverse area of the possible design space. Users can nominate the population size, number of generations for each run and select rooms, zones and houses from any generation in any run as suitable for final room, zone or house populations. These selections are made interactively by users as solutions appear which are judged favorable, based perhaps on factors not included in the fitness function. Such selections may therefore not be optimal according to the given fitness function[1].

[1] Ibid.

Results are shown in the following figures (Fig. 4.23 – Fig. 4.26) for room, zone and house generation.

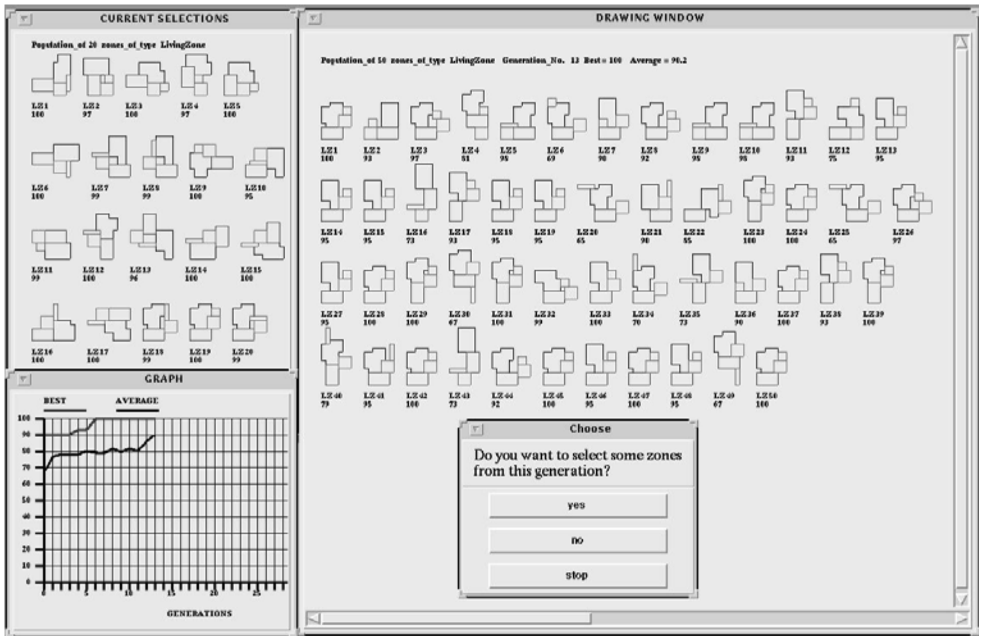


Population_of 5 rooms_of_type LivingRm

Population_of 60 rooms_of_type LivingRm Generation_No. 17 Best = 100 Average = 75.4833

Enter Room Number 1, 41

Fig. 4.23: Results of Living Room Generation after the 17th generation.



Population_of 20 zones_of_type LivingZone

Population_of 50 zones_of_type LivingZone Generation_No. 13 Best = 100 Average = 90.2

Do you want to select some zones from this generation?

Fig. 4.24: Results of Living Zone Generation.

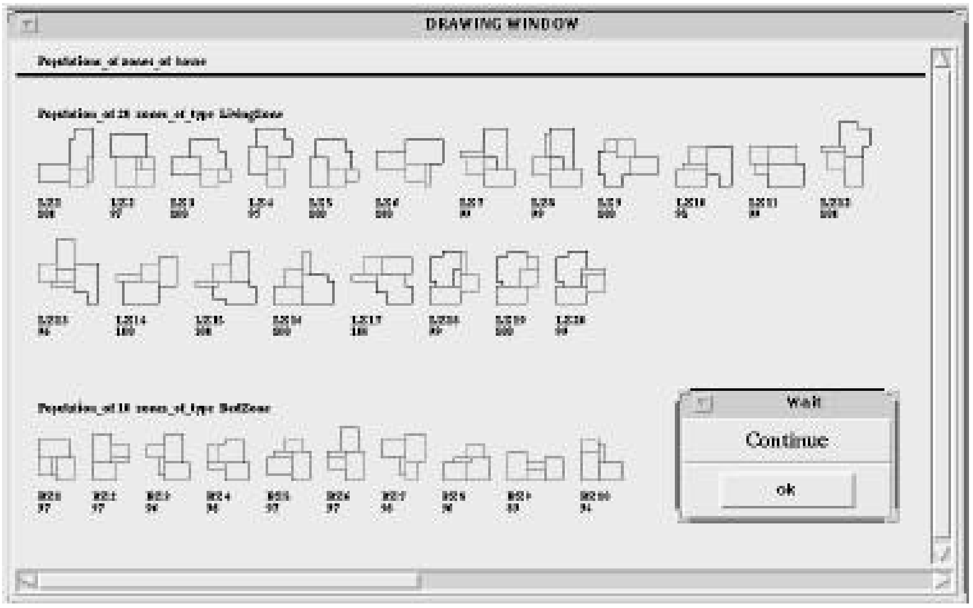


Fig. 4.25: Results of Bed and Living Zones Generation.

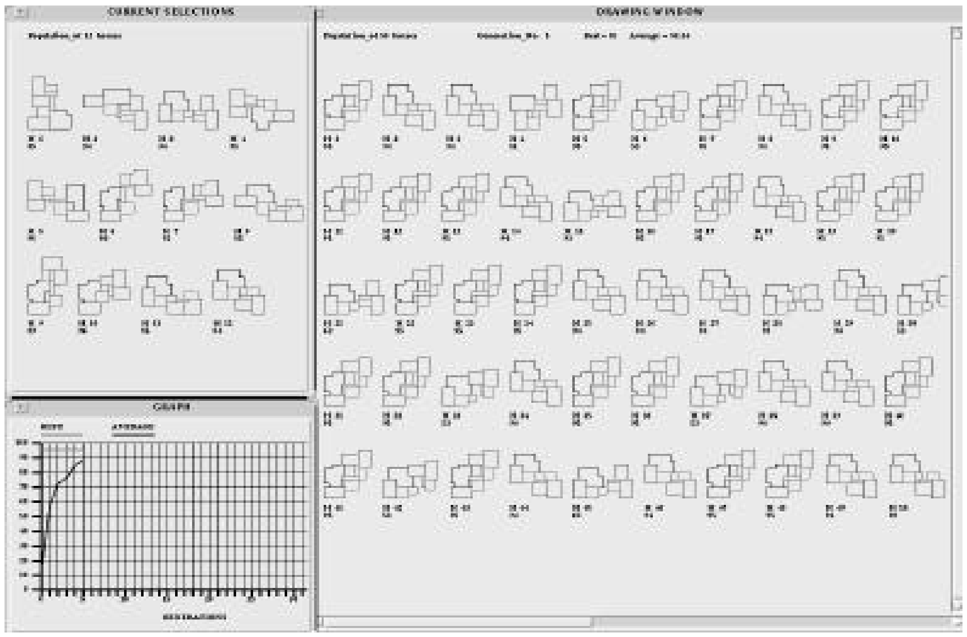


Fig. 4.26: Results of House Generation.

(Fig. 4.23) shows the 17th generation of the evolution of this population of 60 members. A fifth room shape was selected at the 14th generation and two more room shapes (Room Numbers 1 and

41) are being selected. The upper line in the graph shows the evolution of the best solution while the lower line shows the evolution of the population average.

Other rooms were generated in a similar way. The room areas generated were: (a) Living Zone: Living Room 24; Dining Room 15; Kitchen 9; Entrance 4; (b) Bedroom Zone: Master Bedroom 15; Bedroom 12; Bathroom 6; Hall 3. (Fig. 4.24) shows the results of the Living Zone generation. The initial population of 50 Living Zones at run 1 was randomly generated by selecting rooms from the final selections for the Living Room, Dining Room, Kitchen and Entrance. Twenty Living Zones have been selected by the user. (Fig. 4.25) shows the set of Bedroom and Living Zones selected. (Fig. 4.26) shows the final set of houses generated in this example.

4-1-3-2 Generating 3D Forms

Applying the same concept previously stated in 2d forms, multiple architectural 3D forms mate and deliver their features (genotypes) to the next generation. This allows desirable features to evolve independently and later be merged into one single individual. (Fig. 4.28, 4.29) shows various 3d compositions generated using GA system applied on shapes generated with shape grammar system (Fig. 4.27), by selecting 2 individuals (parents) to mate and generate[1].

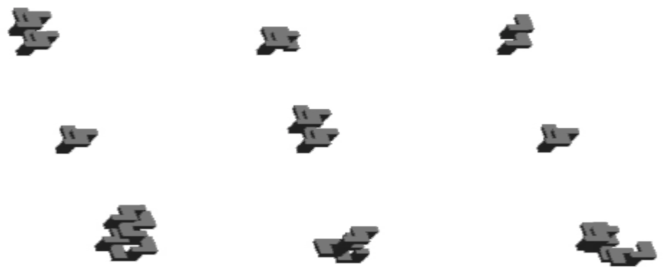


Fig. 4.27: Forms generated using shape grammar system.

[1] Loomis, B.: *A Note on Generative Design Techniques: SGGA a User-Driven Genetic Algorithm for Evolving Non-Deterministic Shape Grammars*. Massachusetts Institute of Technology, Cambridge MA, USA,2000.

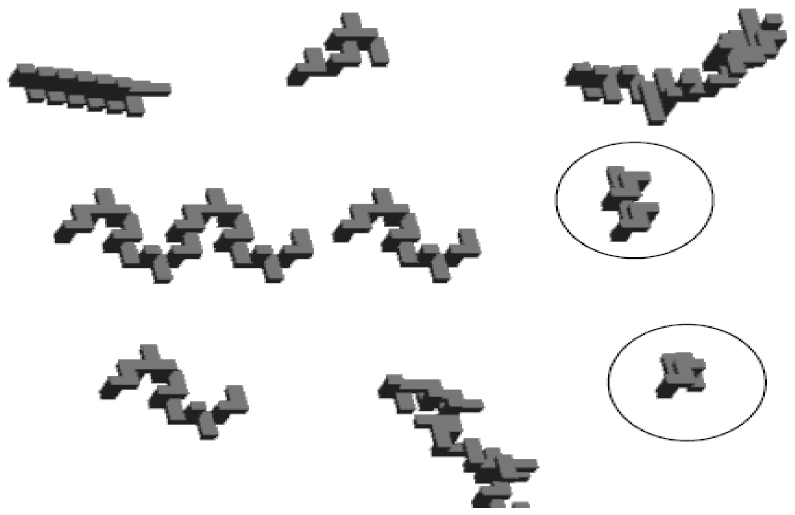


Fig. 4.28: Parents selected for mating and generating more forms.

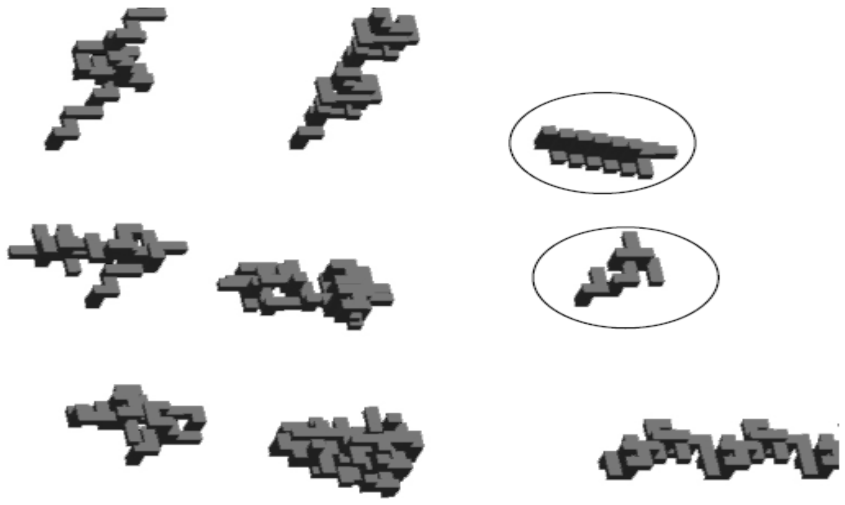


Fig. 4.29: Repeating step for generating more individuals.

The next are four experimental examples, showing how the individuals in each generation of the genetic programming undergo reproduction via the choice of two genetic operations[1]:

i. crossover (sexual recombination)
ii. mutation

[1] Yang, D. and Tang, M.: *Genetic Evolution: A Synthetic Approach in Form Generation*. Savannah College of Art and Design (SCAD), Georgia, USA, http:// genetic.ming3d.com/GE FEIDAD3.pdf. Accessed 16/10/2007.

1- A Roof form generation example (using crossover)

A roof form generation was facilitated using Maya program. A prototype matrix was created to drive the deformation of target geometry mesh. 100 children were produced from each pair of “roof parents”. The first child was identical to parent-A, and the 100th child was identical to parent-B. The other 98 children were just the mixture of parent A and B with the different weight combination. For instance, the second child had 99% affluence from A and 1% affluence from B, the third child had 98% affluence from A and 2% from B, etc. (Fig. 4.30).

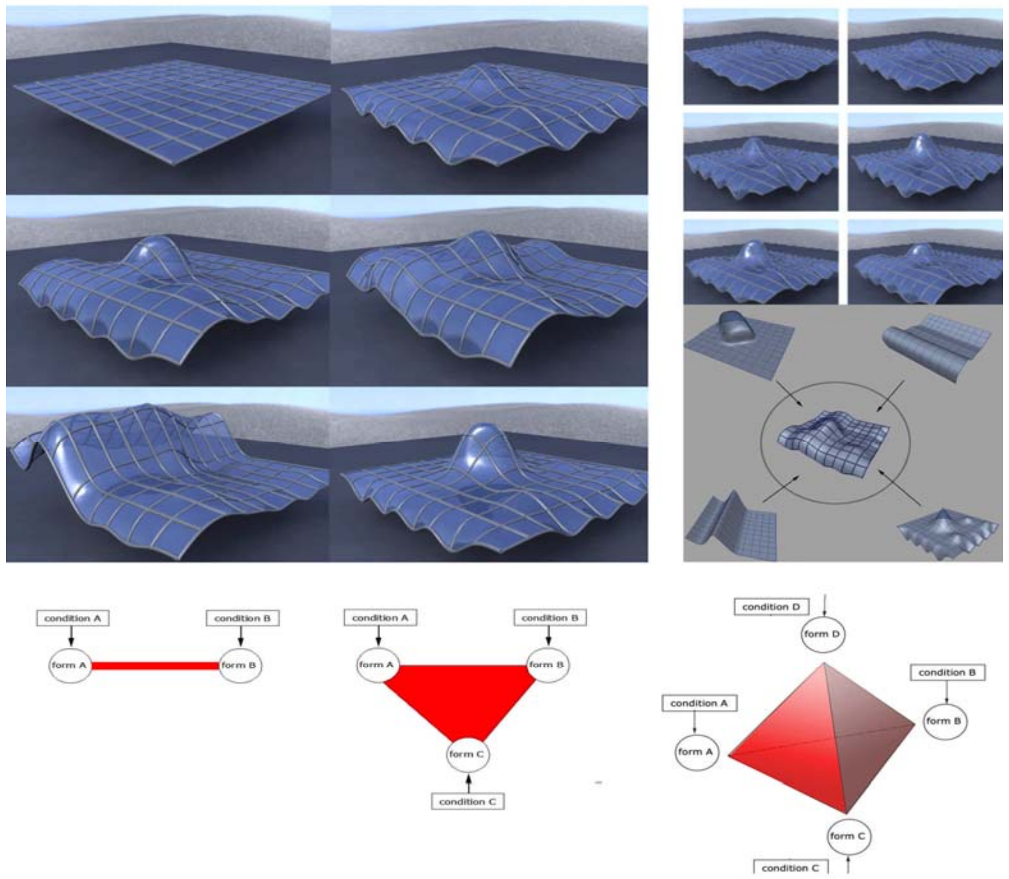


Fig. 4.30: By mating two, three or four successful roof structures, a large quantity of offsprings were generated.

2- A Tower's skin structure example (using mutation)

Inspired by the role of mutation of an organism’s DNA in natural evolution, mutations simulated in one or more members of a tower skin current population, yielding a new candidate solution. There are two strategies to add mutation in this process.

i. Using random noise

In (Fig. 4.31), random noise was added into the procedural model. The noise normally had a subtle effect on the phenotype unless we amplified the result or the noise accumulated several generations. Controlled by the Turtle script and L-Systems in Maya, this type of mutation allowed the complexity of the form to grow continuously as evolution proceeded.

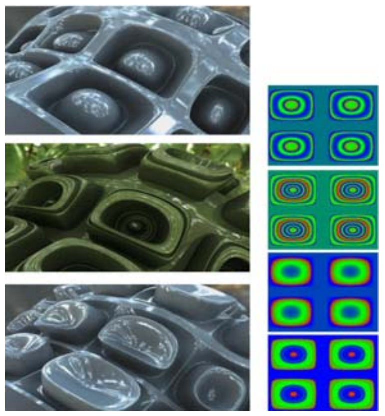


Fig. 4.31: Width and bevel values were mutated by adding weight map.

ii. Using weight map

In (Fig. 4.32), each voxel’s mutation probability was controlled by it’s coincide pixel’s alpha value in a 2D gray scale weight map. The weight map was edited in Photoshop and applied 2D filters such as noise and blur. After a mutation weight map completed, an amplifier was added to its original displacement value and applied to its controlling voxels in the tower’s skin structure.

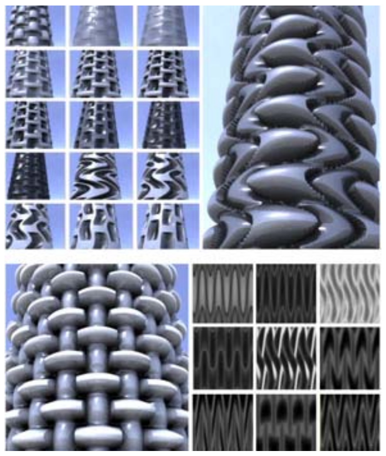


Fig. 4.32: Height, width and bevel values were mutated by adding random noise.

3- Hybrid house example (using crossover)

In nature, when two individuals mate, each parent passes half of its paired chromosomes onto its common offspring. The chromosomes combine to form new pairs, which lead to a unique new individual with phenotypes inherited from both parents. Individuals with more adapted genotypes will survive in the evolution process while others will eventually be eliminated.

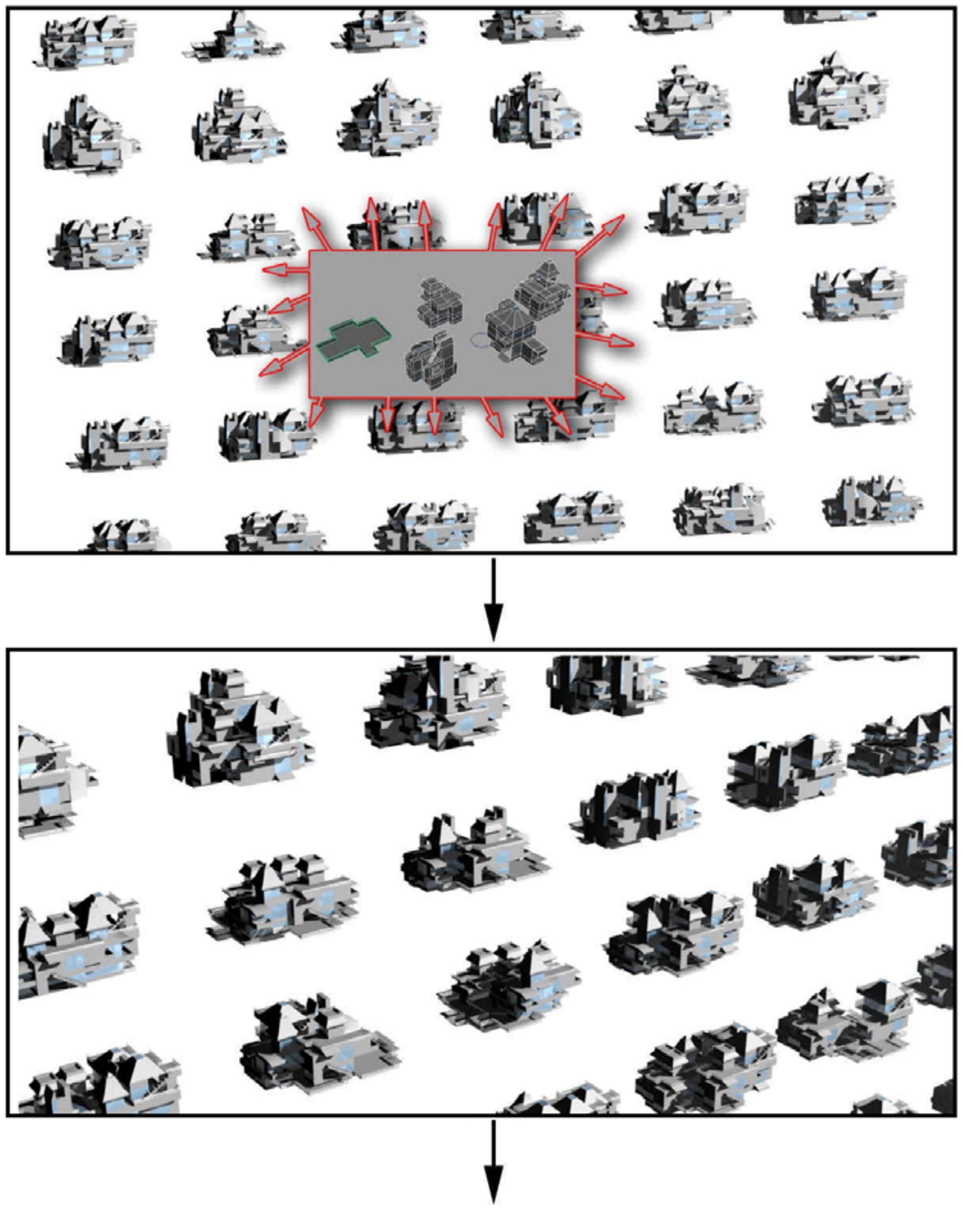
Inspired by this nature analog, five building units were designed using the CAD program and exported them into Maya. Then, a program was written in MEL language to execute GE and produced 3125 offspring in the first generation, by an exhaustive combination of five original units' genotype. From these 3125 samples, only five ideal spatial arrangement solutions were selected by reviewers and then used as the genotype for the next generation.

In the third generation, three nonlinear deformation nodes (bend, twist, and wave) were evolved independently in the evolution and then explicitly added to the five units to yield a more complex layout potential. As a result, a high degree of complexity was generated.

In this process, GE demonstrated itself with great power and unlimited potential of form reproduction driven from sets of genetic parameters. The reviewers selected the desired spatial layouts that survived and reproduced them to create the new generation.

In the fourth generation, a central courtyard was introduced into the evolution as a "void unit" and blended with the selected layouts. Another input variable, time, as the 4th dimension, was also added to freeze all the layout possibilities into a motion.

Expressions were evolved and various spatial arrangements were produced as the value of time was smoothly animated. All of the 3125 generated housing models were captured into a single morphing animation (Fig. 4.33).



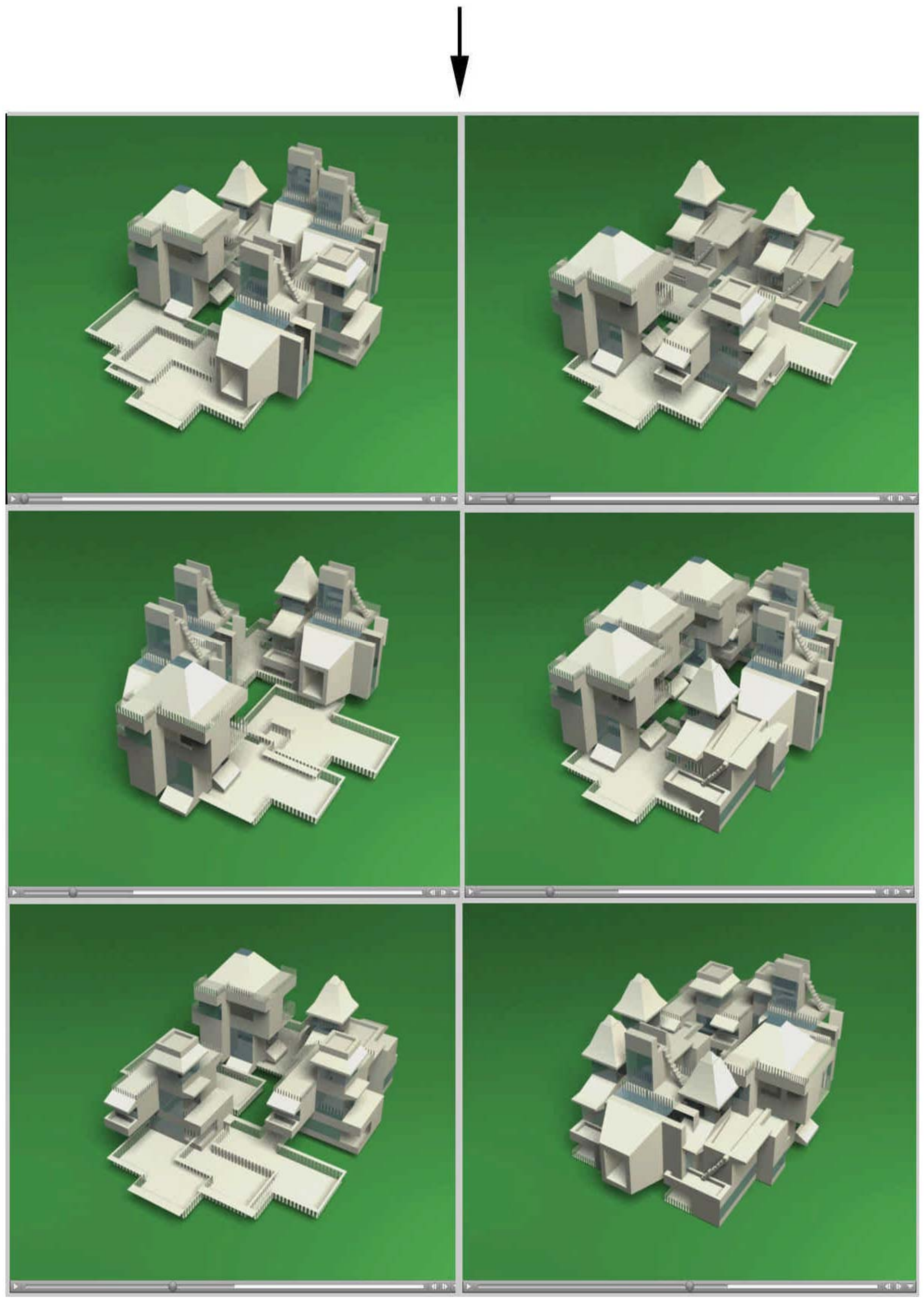


Fig. 4.33: The animation was captured from 3125 spatial arrangement solutions crossed four generations.

4- A Chameleon example (using mutation)

An experimental project introduces a new approach to design a shelter form, a filter to the tropical climate that responded to the changing environment like a chameleon skin.

First, a Non-Uniform, Rational, B-spline (NURB) surface was subdivided by 36 driving nodes in Maya. These nodes allowed the surface to be dynamically configured in real-time. Then, 2D displacement maps were created to shape the feature of each node and sculpted a series of 3D surfaces by the process of mapping and morphing. Two parameter sets were defined. Each of them could create an ideal building skin with different successful features, such as climatic and topographic responsiveness. It became desirable to combine these two features into one single skin structure (Fig. 4.34).

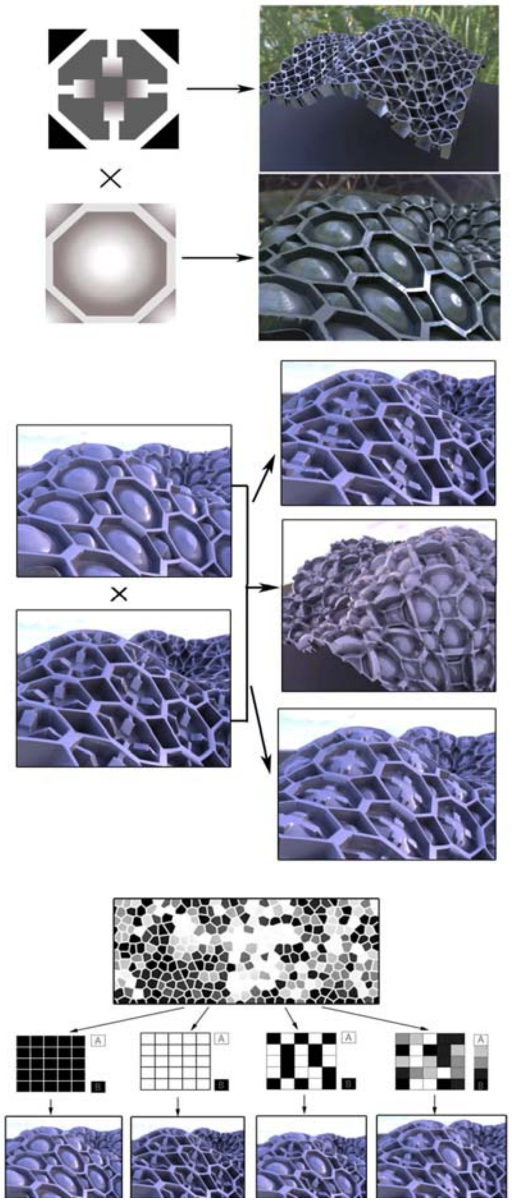


Fig. 4.34: As the chameleon skin, a gray scale weight map was projected to the NURB surface.

The new skin generated according to the weighed maps suggested a new solution that could constantly change its surface corresponding to the changes of these environmental conditions.

4-2 Cellular Automata

4-2-1 Introduction

4-2-1-1 Definition

Cellular automata are a computational method simulating the process of growth and evolution by describing a complex system by simple individuals (cells) following simple rules[1].

4-2-1-2 Outlines

Concept of simulating growth was introduced by John von Neumann[2] and further developed by Ulam[3] in the area of simulating multi-state machines. The concept gained greater popularity when Martin Gardner[4] described John Conway's "Game of Life", a game that generated two-dimensional patterns. Stephen Wolfram[5] in the eighties began researching the concept to represent physical phenomena.

4-2-1-3 The Evolutionary Concept

Cellular automata viewed as a mathematical approach differs from traditional deterministic methods in that current results are the basis for the next set of results. In the recursive methods the outcome usually can not be easily anticipated. This offers an interesting and rich platform from which to develop possible architectural patterns. This recursive replacement method continues until some state is achieved.

---

[1] Krawczyk, R.J.: *Architectural Interpretation of Cellular Automata*. Illinois Institute of Technology, USA, Generative Art 2002.

[2] Von Neumann, J.: *The General and Logical Theory of Automata*. In J. von Neumann, Collected Works, edited by A. H. Taub, 1963.

[3] Schrandt, R. and Ulam, S.: *On Recursively Defined Geometrical Objects and Patterns of Growth*. In A. Burks (Ed), Essays on Cellular Automata, University of Illinois Press, Urbana, pp. 232-243, 1970.

[4] Gardner, M.: *The Fantastic Combinations of John Conway's New Solitaire Game of "Life"*. Scientific American, 223, pp. 120-123, 1970.

[5]Wolfram, S.: *A New Kind of Science*. Wolfram Media Press, Champaign, 2002.

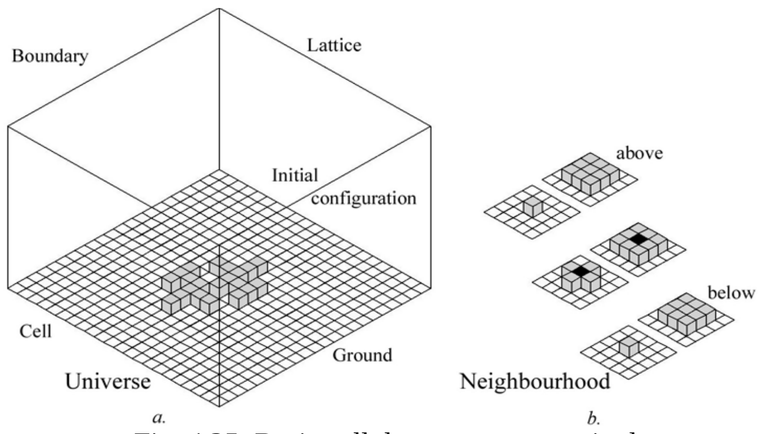


Fig. 4.35: Basic cellular automata terminology.

The three-dimensional universe (Fig. 4.35 a) of cellular automata consists of an unlimited lattice of cells. Each cell has a specific state, occupied or empty, represented by a marker recording its location. The transitional process begins with an initial state of occupied cells and progresses by a set of rules to each succeeding generation.

The rules determine who survives, dies, or born in the next generation. The rules use a cell's neighborhood to determine its future. The neighborhood can be specified in a number of ways, (Fig. 4.35 b) displays two common methods of determining which adjacent cells to consider.

The rule developed by Conway is: check each occupied cells' neighborhood, survival occurs if there are two or three neighbors, death occurs if there are any other number of neighbors, and birth occurs in an empty cell if it is adjacent to only three neighbors. As each generation evolves, one of four cases can occur over some period of time. Either the cells find a stable form and appear not to change; or they become what is called a "blinker" and alternate between two stable states; or all or a cluster of the cells become a "glider", a group of cells that begins to transverse the universe forever, or all the cells die, extinction. A variety of rules have been proposed, with Conway's being the starting point[1].

[1] Krawczyk, R.J.: *Architectural Interpretation of Cellular Automata*. Illinois Institute of Technology, College of Architecture, USA, Generative Art 2002. op.cit.

4-2-2 Architectural Potentials

The connection to architecture is the ability of cellular automata to generate organized patterns. From this organized patterns it might be able to suggest architectural forms.

The pure mathematical translation of cellular automata into architectural form includes number of issues that do not consider built reality. For example, (Fig. 4.36 a) displays an initial configuration and its raw results at the 8th generation (Fig.4.36 b).

The interpretation or translation to a possible built form can be dealt with after the form has evolved or it can be considered from the very beginning. Deciding to follow a combination of both approaches, a boundary is placed on the lattice to represent a site, along with a ground plane, and an orientation of growth that is vertical and to the sides, but not below. The cells are stacked over each other to create a vertical connection without a vertical displacement between layers of cells[1].

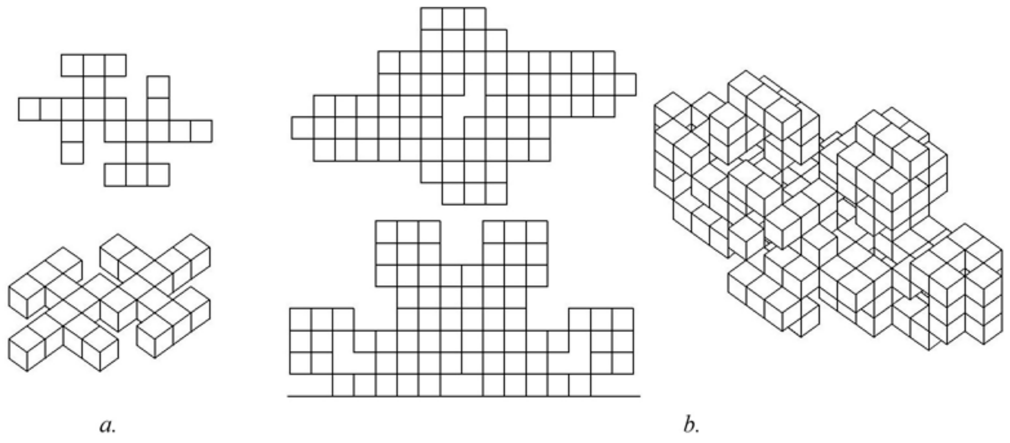


Fig. 4.36: Sample generation.

An initial review of the results highlighted a number of other issues, some cells were not connected horizontally to others and some cells had no vertical support.

[1] Ibid.

4-2-2-1 Generating 2D Forms

A series of interpretations were made by Krawczyk[1] to address the horizontal connection of cells resulting 2D architectural spaces. The centroid of each cell becomes the basis for this horizontal development (Fig. 4.37).

(Fig. 4.37 a) displays the initial cell configuration at a typical layer, each cell is adjacent to another. Cells are first joined together to form the largest contiguous floor areas possible. In this configuration, the cells that are diagonally adjacent do not connect horizontally.

(Fig. 4.37 b) the cell remains a square unit but is scaled so to overlap its neighbors. When joined, a small connector at the diagonals appears.

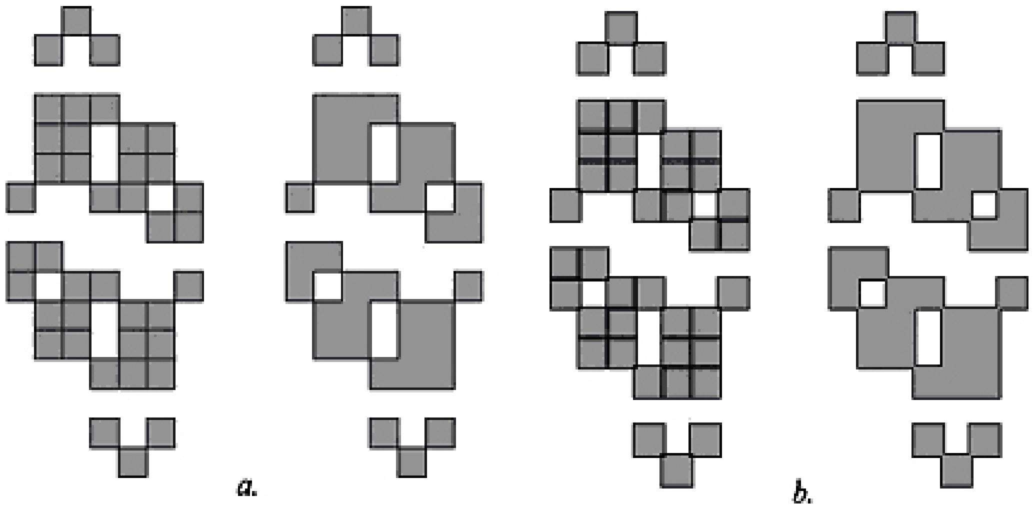


Fig. 4.37: Horizontal connection of cells.

[1] Ibid.

In (Fig. 4.38) the scale of the square unit is increased to further develop a connector. The entire character of the exterior edge of the initial cells changes by these interpretations, as well as, addressing the interior horizontal connections between unit spaces. Additionally, a series of interesting interior openings begin to emerge.

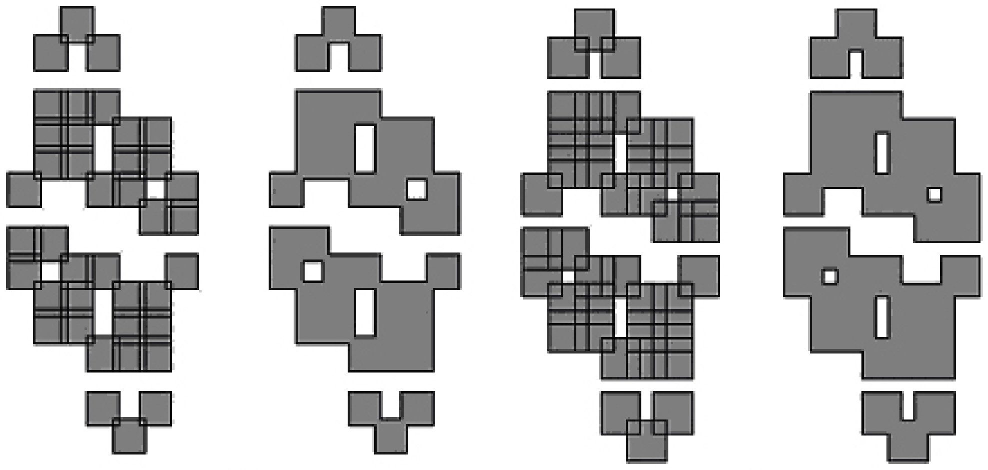


Fig. 4.38: Horizontal connection of cells with increasing scale.

In addition to a square unit, a variety of other shapes could be investigated that would articulate the building edge in other ways than the square and that could accommodate orientation and additional surface area in elevations for fenestration.

(Fig. 4.39) displays a series of possible unit shapes: circular, super ellipse, rotated square, and a hexagon.

The joining of the units' spaces, in addition to creating large contiguous areas, also creates a series of edge points, an envelope that can be further given an interpretation or transformed.

This envelope can be interpreted as a series of curve segments or a spline (Fig. 4.40). Depending on the type of unit shape, a variety of curved edges begin to develop.

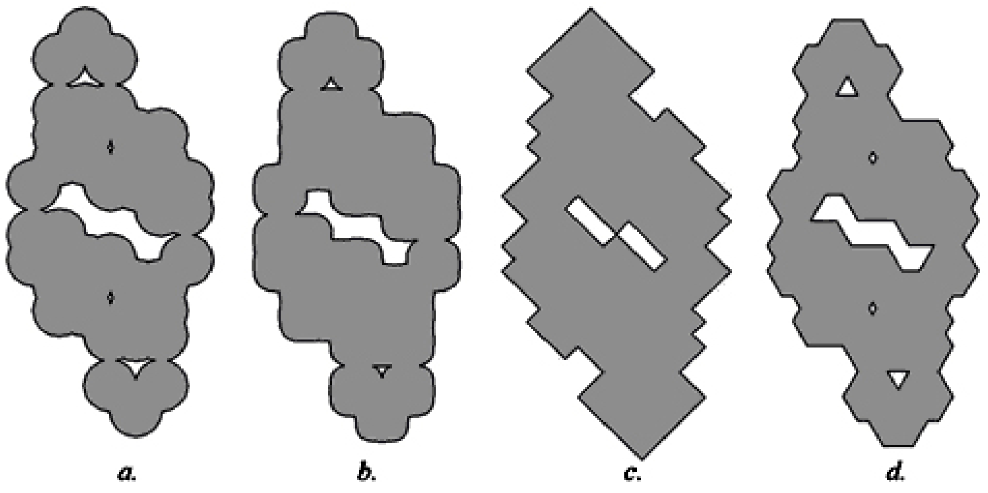


Fig. 4.39: Variation of unit shape.

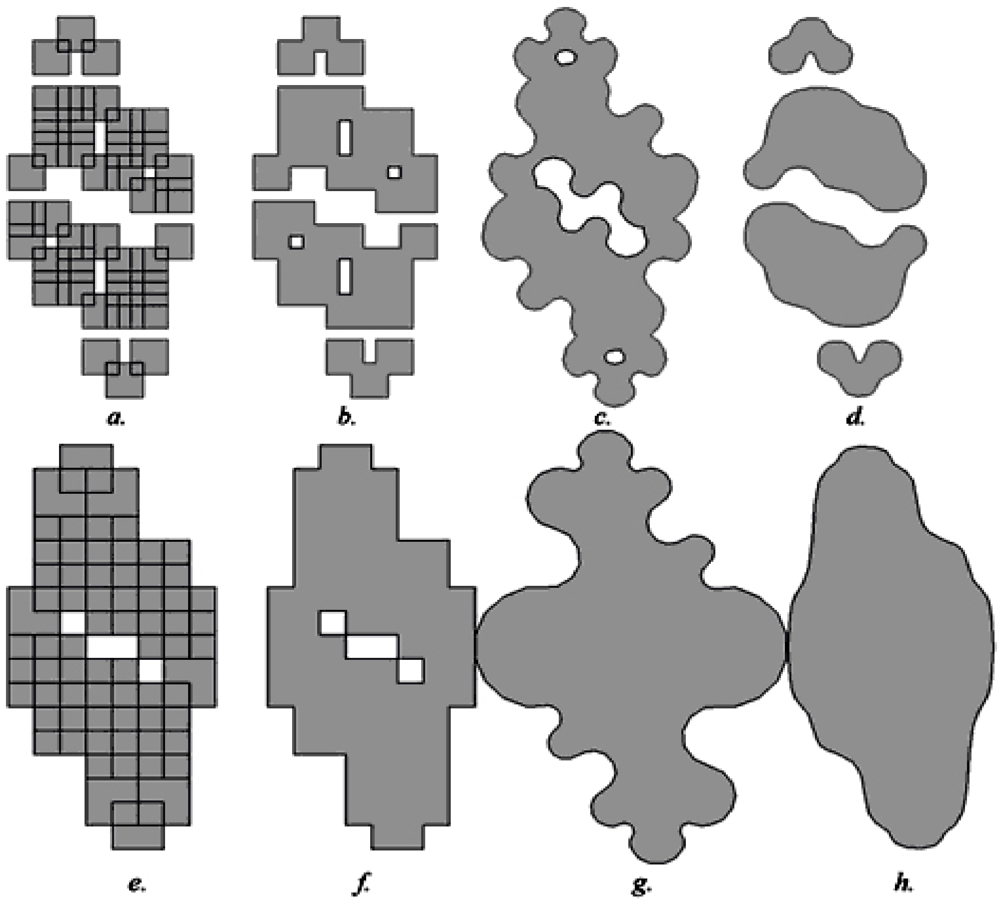


Fig. 4.40: Envelope interpretation.

4-2-2-2 Generating 3D Forms

An early example of three-dimensional pattern development is the wooden block model created by Schrandt and Ulam (Fig. 4.41 a). Bays Investigated repeating patterns as Conway had found in two-dimensions (Fig. 4.41 b). And finally a highly inspirational architectural application by Coates (Fig. 4.41 c), which had the same spirit as Bays. The most recent is two methods develop by Wolfram (Fig. 4.41 d), in which a stacking method is explored, as well as, one similar to Bays. The striking similarity in these is the explicit representation of the cellular automata, even though each had taken a different approach and had a different application as an investigative goal.

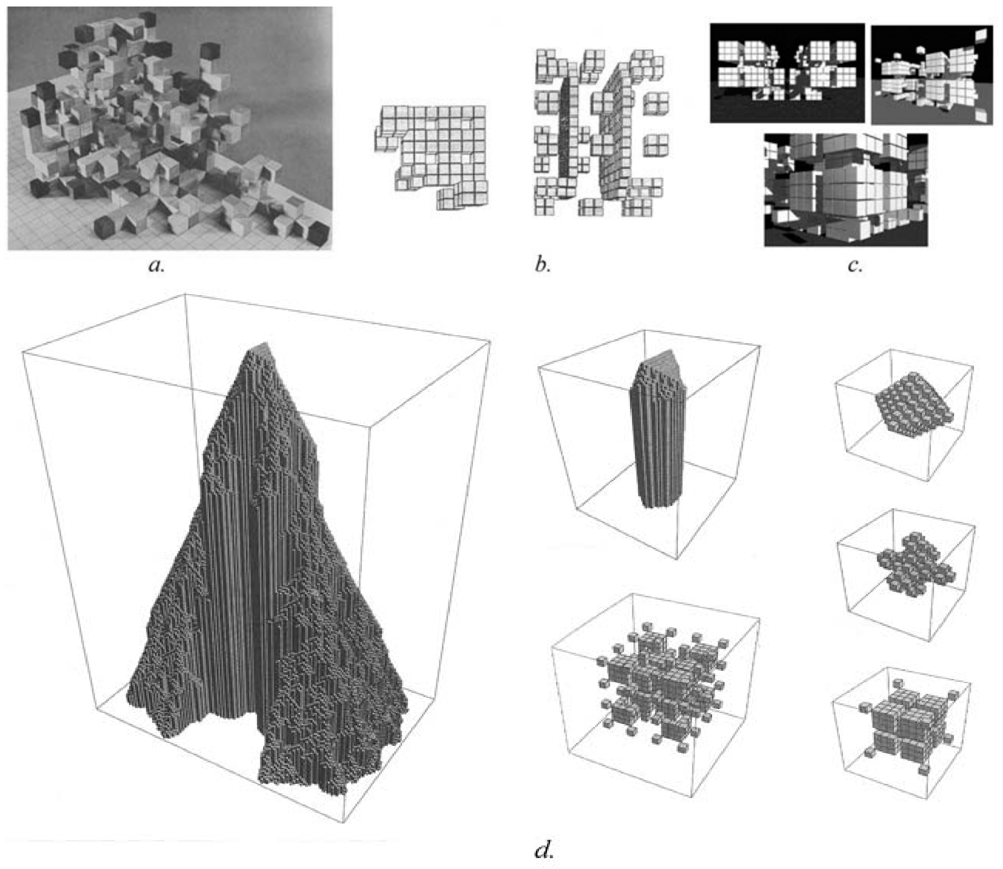


Fig. 4.41: Early three dimensional CA patterns examples[1].

[1] Krawczyk, R.J.: *Experiments in Architectural Form generation Using Cellular Automata*. Illinois Institute of Technology, College of Architecture, USA, (design e-education) Modeling Real and Virtual Worlds Session 15, eCAADe 2002.

Krawczyk developed many experiments on using CA in architectural design. He interpreted the generated abstract form to three dimensional architectural forms in many ways, like:

- **Adding vertical supports**[1]

In 2D growth, the initial cell configuration lacked in having vertical supports. This issue could be addressed in the growth rules by limiting growth that had cell supporting it from below or to add supports to the final configuration. (Fig. 4.42) displays two possible support strategies, one with columns at the each cell corner and the second, columns located at the center of each cell.

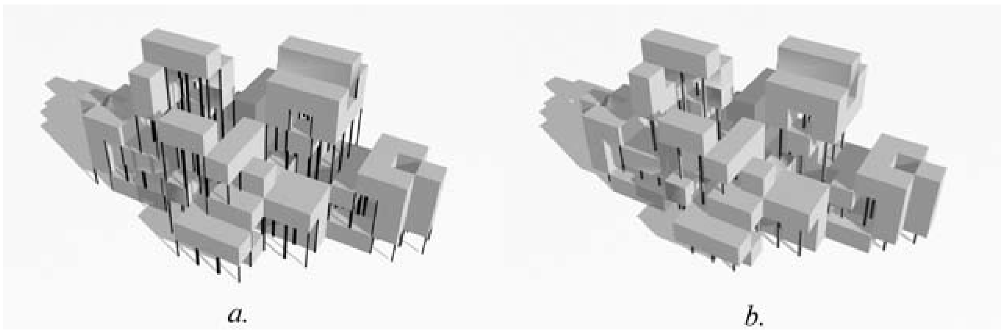


Fig. 4.42: Cell supports.

- **Cell growth variations**[2]

(Fig. 4.43 a) is the raw cell configuration with supports represented as a mass model and with the cells represented as spatial modules of three floors each. Individual floor plates are included and each set of merged cells has a glass enclosure.

(Fig. 4.43 b, 4.43 c) are the curve and spline versions. One of the interesting aspects on this particular interpretation is the interior spaces created by the merging of the cells.

[1] Krawczyk, RJ.: *Architectural Interpretation of Cellular Automata*. Illinois Institute of Technology, College of Architecture, USA, Generative Art 2002. op.cit.

[2] Ibid.

A number of other merge schemes were investigated to further develop this concept. To articulate the edges of each layer of cells, a variety of spatial units, as shown previously, were also investigated.

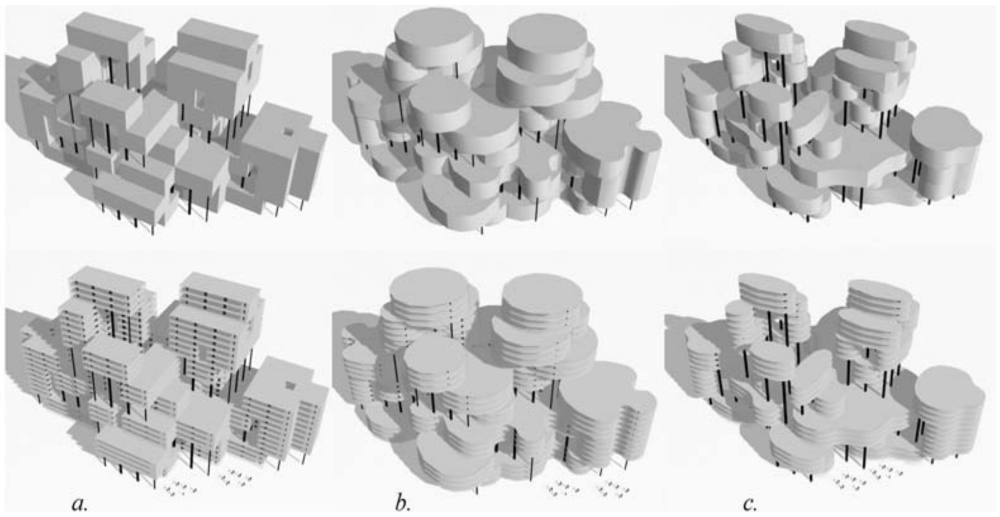


Fig. 4.43: Basic architectural form series.

- **Different interpretations**[1]

Other approaches to the interpretation of the unit cells were also investigated. (Fig. 4.44) highlights an approach where the size of the unit cell is given a minimum and maximum, the actual size is selected randomly. The random method was also implemented in (Fig. 4.45), a minimum and maximum offset was defined for each vertex of a cell, then selected randomly. The shape in both of these cases remains approximately the same to the original.

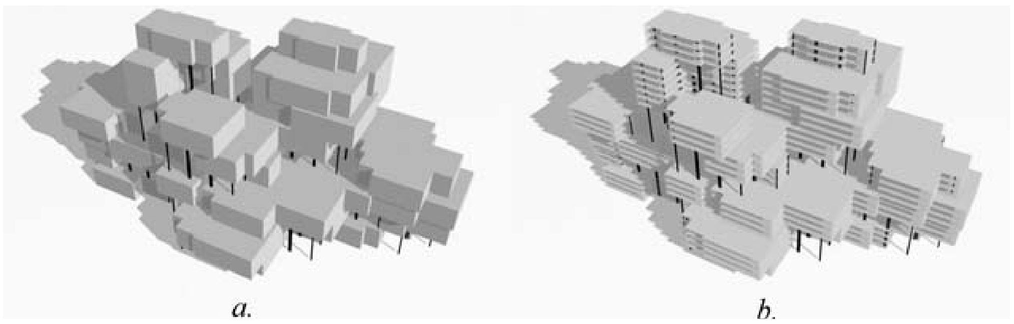


Fig. 4.44: Cells of random size.

[1] Ibid.

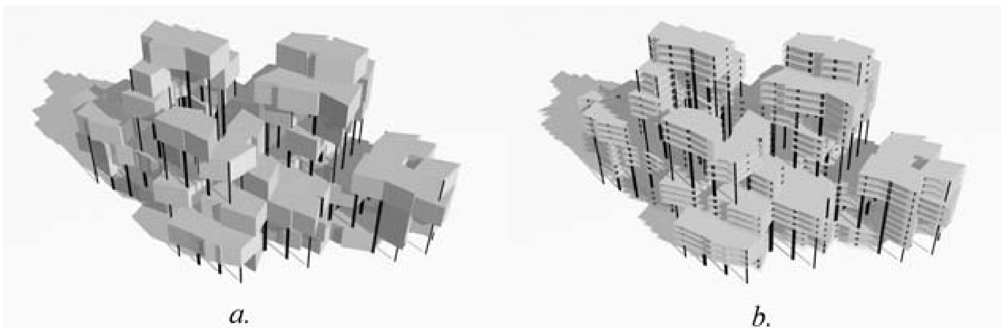


Fig. 4.45: Cells with random offset of vertices.

An entirely different approach was also investigated in that the vertical aspect of the stacked cells was considered as primary after the basic horizontal connections were made. (Fig. 4.46) displays one such example using the square cell unit.

The final concept considered was to interpret the cell formations as they are created. In this case, called retained growth, in each generation when a cell survives, it increases in size. This approach considers the actual growth process in the cellular automata and interprets it directly. (Fig. 4.47) display such a example.

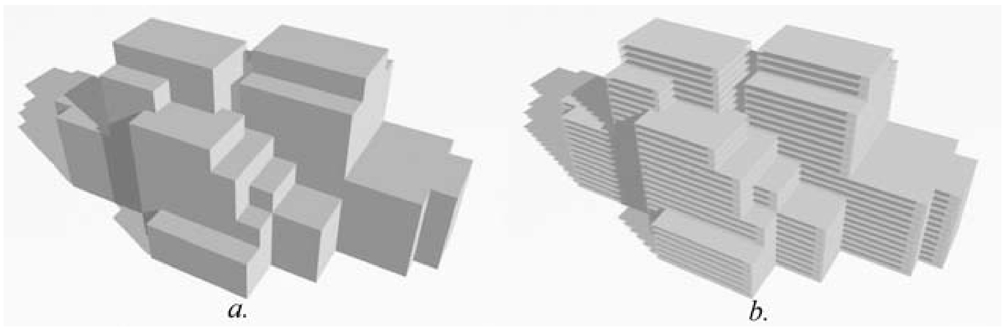


Fig. 4.46: Cells as vertical volumes.

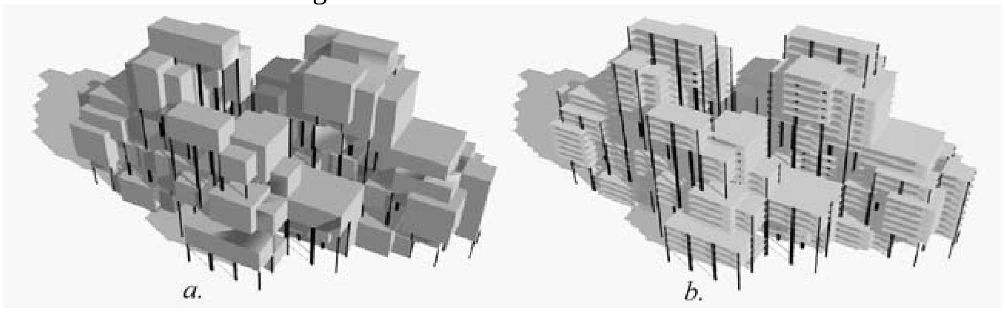


Fig. 4.47: Cells with retained growth.

- **Skinned with an envelope**

Still other methods which have been developed by others, offer possibilities for future investigations. One in particular was suggested by Coates[1], in which the entire three-dimensional cell configuration is skinned with an envelope (Fig. 4.48). The challenge would be to use this method but still embed the floor and unit space concept that was developed in this paper.

The varieties of methods on interpretation are only limited by the actual mathematics of the generating concept, the ability of the tools we use to model it, and our imagination.

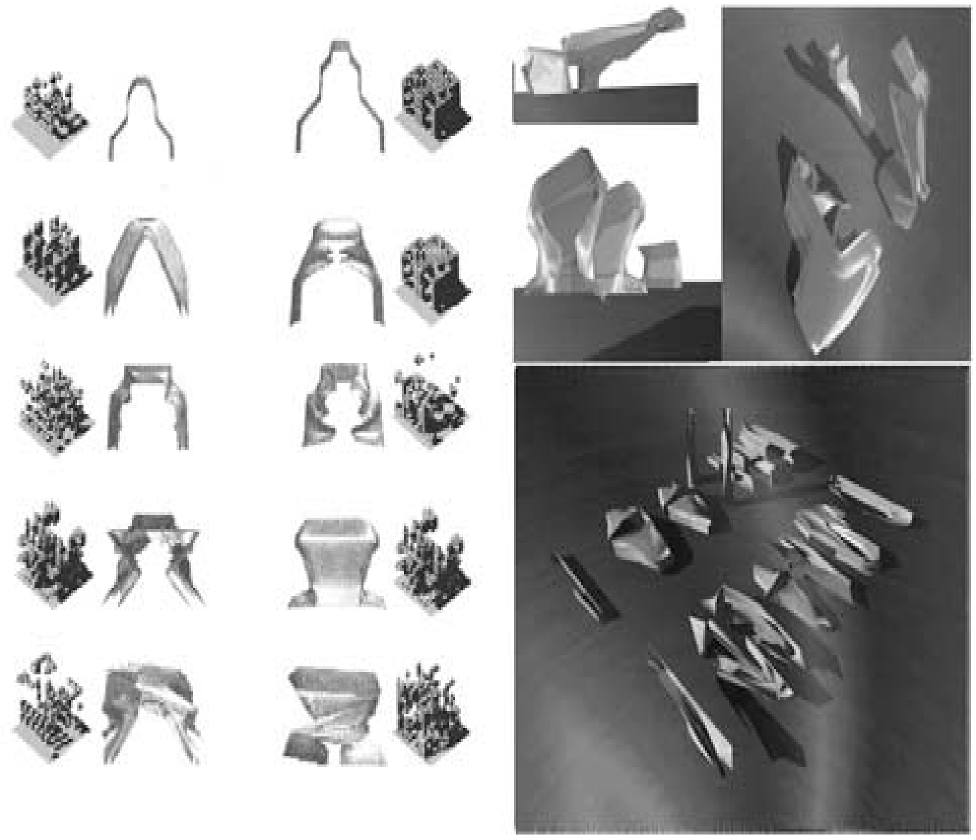


Fig. 4.48: CA forms Skinned with an Envelope.

[1] Coates, P.: *The Use of Cellular Automata to Explore Bottom Up Architectonic Rules*. Eurographics Conference, Imperial College of Science and Technology, London, 1996.

Summary of chapter four:

Chapter four discussed another two generative tools evolved when computer science reciprocated with evolutionary biology of natural systems that had been proposed by Charles Darwin's theory of evolution to produce evolutionary computation. Then evolutionary computation in turn reciprocated with design to evolve evolutionary design. This chapter covers two generative tools belonging to evolutionary design like: Genetic algorithms and cellular automata. The chapter Studied: their nature, mathematical concepts and architectural potentials, with the purpose to use them as form generators in the preliminary phase of the form finding process.

Chapter 5
Discussions, Results and Recommendations

Chapter 5: Discussions, Results and Recommendations

5-1 Discussions

- **In the previous chapters we discussed:**

-WHAT are the tools that one can use in order to generate forms?
-WHAT is the Evolutionary & Mathematical concept of each?
-HOW Mathematics has been shifted from being a supporting tool into a generative medium?
-HOW Evolutionary and Mathematical elements can be interpreted into elements with architectural potentials?

- **While in this chapter we will discuss:**

-WHAT are the characteristics and promises of these tools?
-HOW some architects apply the results of these generative tools in forming their design basis?
-WHEN each of these generative tools is efficient?
-WHY should we use these generative tools in design?

First there are two different methods to design form with computers[1]:

1- The ComputerSupported form Process:

Which intends the possibilities of feedbacks and intersections between man and machine.

2- The ComputerGenerated form Process:

Which is characterized by an automatic process, without the intervention of the user, who only predefines parameters and algorithms for a system. Then filter, select, and evaluate the generated form.

[1] Dürr, C.: Morphogenesis, *Evolution of Shape*. ETHZ, CAAD, NDS 2004.

According to John Gero and Mary Lou Maher, creative design while highly valued is not currently supported by computer programs used during design. Research in design and artificial intelligence shows potential for improving our understanding of design and establishing computational models for creative design. Research in computational processes for design has been primarily concerned with routine design which has not provided much insight into the adequacy of these processes to support creative design[1].

1- The routine design

Is the result of making design decisions in the context of a design situation where all the decision variables are known priori. Thus, in routine design the designer operates within a defined, closed state space of possible designs.

2- The Creative design

Occurs when new design variables are introduced in the process of designing. Thus, in creative design the designer operates within a changing state space of possible designs, which increases in size with the introduction of each new variable (Fig. 5.1).

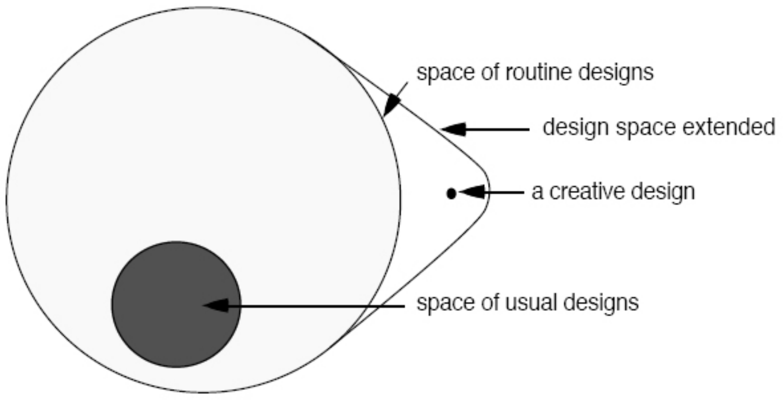


Fig. 5.1: State space for routine and creative design

1 Gero, J.S. and Maher, M.L.: *Mutation and Analogy to Support Creativity in Computer Aided Design*. Design Computing Unit University of Sydney, NSW, 2006, Australia, CAAD futures Digital Proceedings 1991

1- Traditional Design Approach[1]:

In usual traditional design, the role of the designer is to explore a solution space. The key relationship between designer and product is a direct one (even if mediated via a third-party or medium). There is a direct relationship between the designer's intentions and that of the designed product (Fig. 5.2).

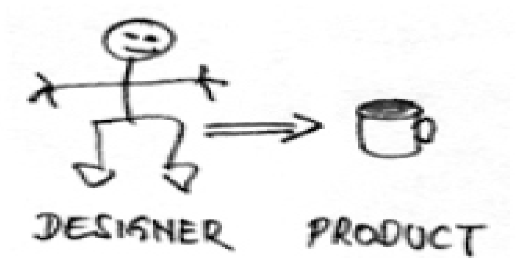


Fig. 5.2: Traditional design approach.

In contrast, design using generative methods involves the creation and modification of rules or systems that interact to generate the finished design autonomously (Fig. 5.3).

2- Generative Design Approach:

Is a design methodology that differs from other design approaches that during the design process the designer does not interact with materials and products in a direct (hand-on) way but via a generative system. Generative design is capable of producing surprising and unpredictable results.

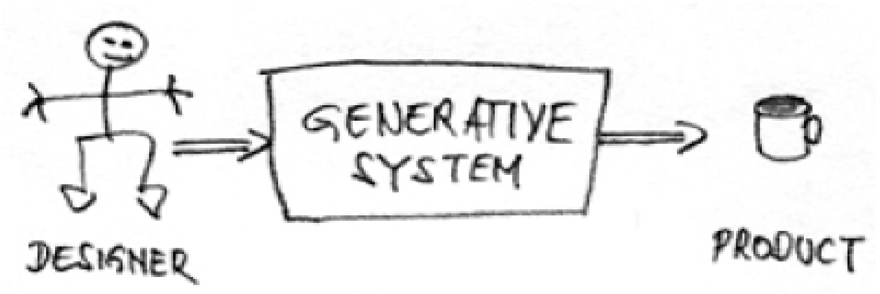


Fig. 5.3: Generative design approach.

[1] McCormack, J., Dorin, A. and Innocent, T.: *Generative Design: A Paradigm For Design Research*. In Redmond, J. et. al. (Eds) Proceedings of Futureground, Design Research Society, Melbourne. (2004).

Thus we can conclude that computer generated form process or the generative design approach introduce new variables into the computer supported form process or the traditional design approach that extended the space of routine design into more creative design space.

The idea that generative systems can support a creative process appears questionable at first glance, as generally known computers are pretty dump and only perform what they are programmed to perform. However, the automatic permutation of large numbers of design elements can indeed inspire ideas and concepts, which designers wouldn't have considered without the support of a generative tool.

The confirmation of the previous statement appears in the sources of inspiration and the essences and theories of creativity all over the decades. These theories were based on: Intuition, unpredictability, metaphorization and no logic. Theory of incubation elaborated by (Wallas), Genploration (Finke, Ward and Smith), Redundant generation (Lem), and Synectics (Gordon). All these theories emphasize the role of unpredictability and metaphors in creativity. Process of metaphorization is characteristic for our era and plays important role in the creative process. Three examples (Fig. 5.4, 5.5 and 5.6) show how one can use metaphorization of abstract forms in the form finding process:

Forms inspired by metaphorization of abstract dynamic forms

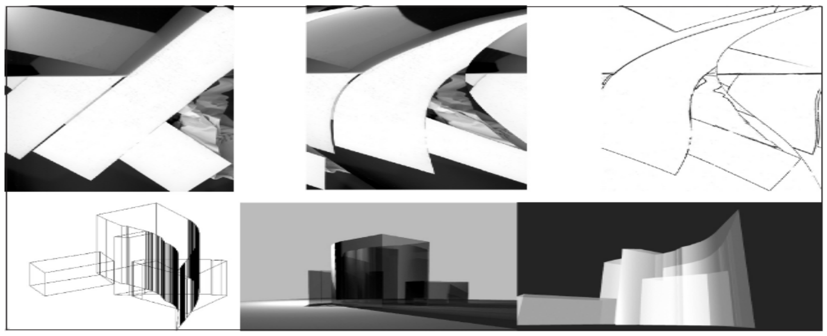


Fig.5.4: Process of form finding using metaphorization of folded paper & cardboard

Forms inspired by metaphorization of nature

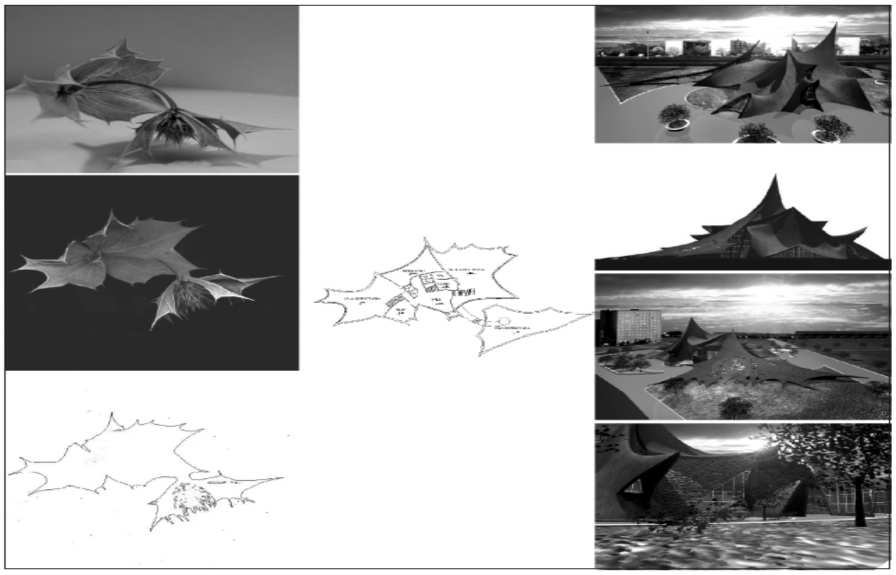


Fig. 5.5: Process of form finding using metaphorization of a sea-holly leaf.

Forms inspired by metaphorization of art

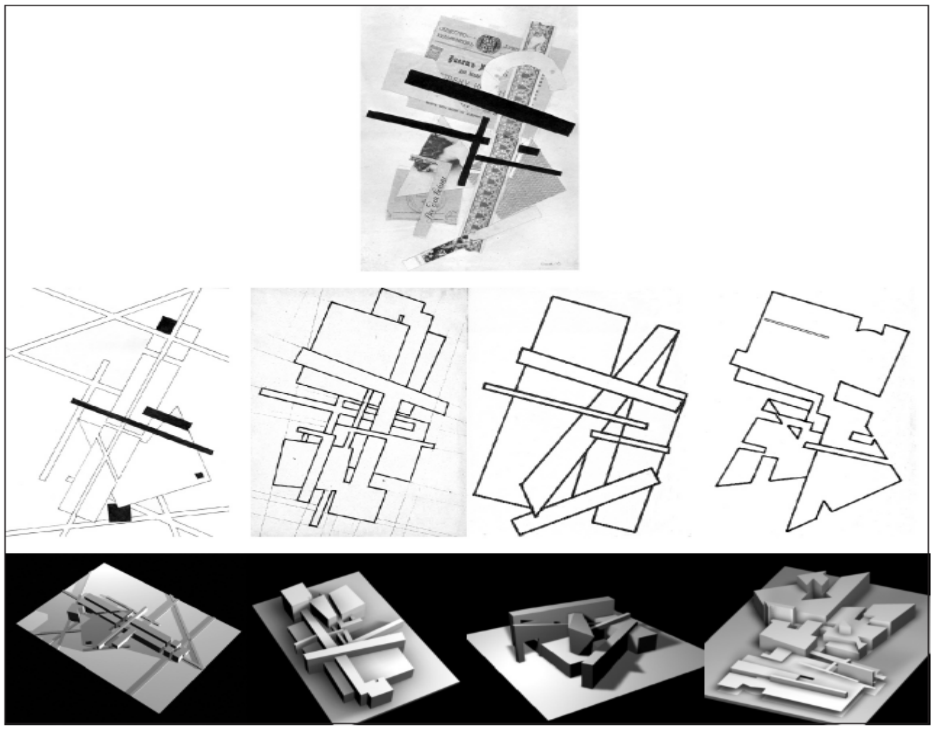


Fig. 5.6: Process of form finding using metaphorization of Rodtchenko's picture.

- **Applications in Architecture**

In recent years, the diversity and complexity of the generated forms using generative systems have turned many artists and architects to these tools to form their design basis, and use generated forms as a trigger for inspiration and creative ideas. The following part will discuss the application of each of these generative tools in architecture forming.

- **Shape Grammars:**

Shape grammars have been used to analyze historical architecture such as Palladian villas and Victorian windows, and to create novel designs such as for Alavaro Siza's Malagueira housing project (work by dr. José Duarte at MIT). Shape grammars are most useful when confined to a small, well-defined generation problem such as housing layouts. A shape grammar can quickly contain a lot of rules, for example the Palladian villas shape grammar presented by William Mitchell contains 69 rules.

Randomly transforming the left hand side initial shape (Shape & Label) to the right hand side through the rule given with repositioning the label and repeating this rule with a certain number, generate derivations that might suggest architectural spaces as shown in (Fig. 5.7).

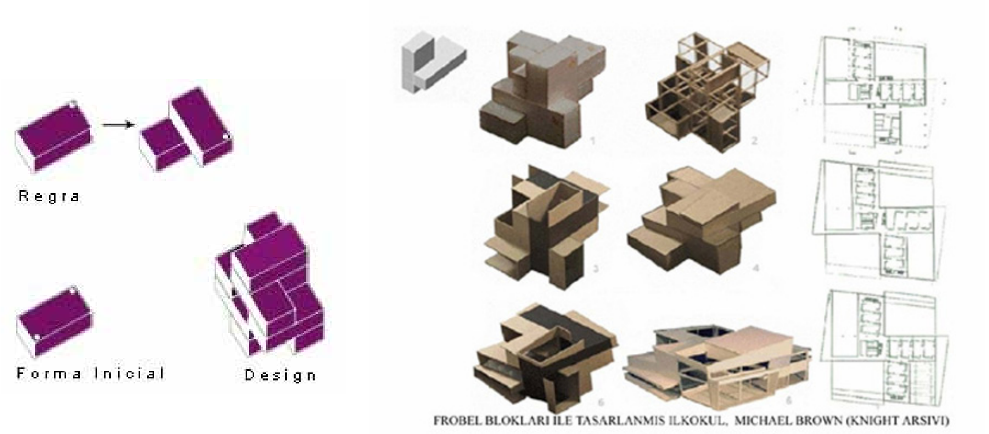

Regra
Forma Inicial
Design
FROBEL BLOKLARI ILE TASARLANMIS ILKOKUL, MICHAEL BROWN (KNIGHT ARSIVI)

Fig. 5.7: (Right)Suggested architectural forms inspired from a shape grammar (Left).

- **Parametric Variations:**

Parametric design can represent complex curves and ruled surfaces in a set of principles encoded as a sequence of parametric equations. Ben Van Berkel used Mobius band resulted from a parametric function as a spatial circulation for the design of Mobius house. (Fig. 5.8) shows his diagram of spatial circulation for Mobius house concept design.

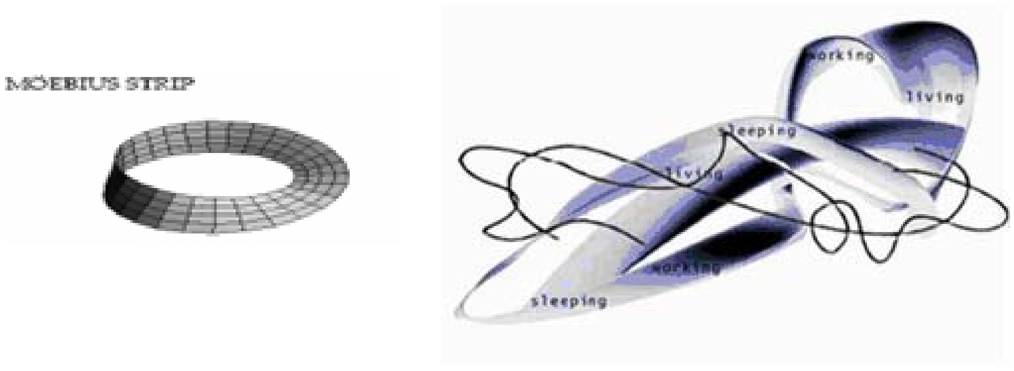


Fig. 5.8: Mobius house spatial circulation Diagram, based on Mobius band (Van Berkel, 1998).

Eight twisted belts are working as Mobius flat surfaces that shaping the Mobius house design. These flat Mobius surfaces are treated as walls, ceilings, and floors that are interlinking inside spaces, (Fig. 5.9, Right) with surrounding exterior, (Fig. 5.9, Left) creating a spatial twist through out the house.

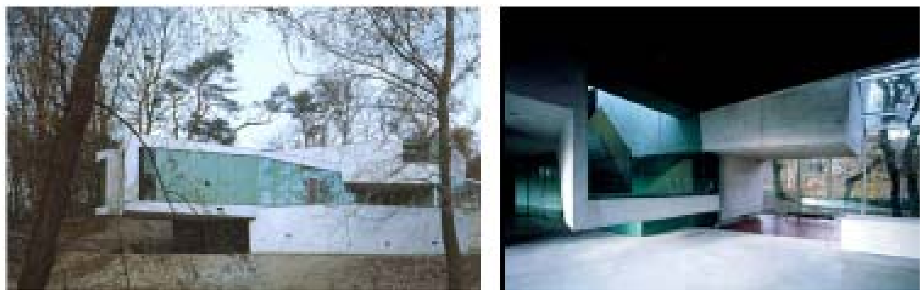
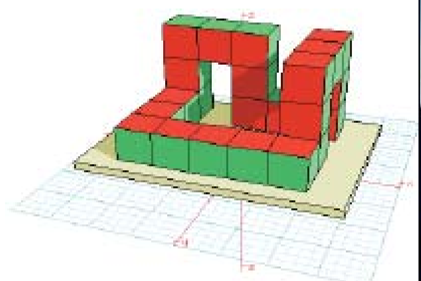


Fig. 5.9: Left: Mobius house exterior, Right: Mobius house interior. (Van Berkel, 1998).

The Office for Metropolitan Architecture (OMA) used Mobius concept for the design of China Central Television (Fig. 5.10).



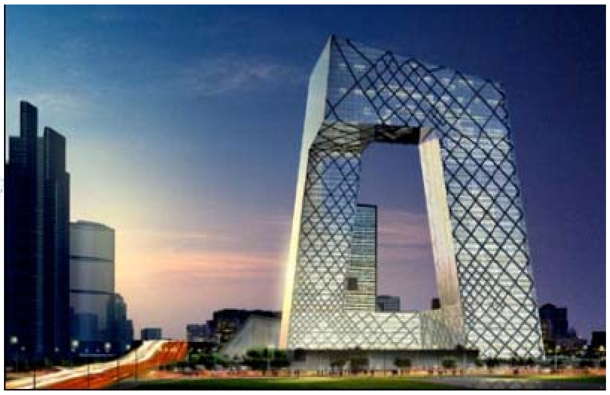


Fig. 5.10: (right) Mobius structure made of cubes, (left) Headquarters for China Central Television (CCTV) in Beijing, China, Scheduled for completion in 2008.

Avant-garde architects like Peter Eisenman and Ben Van Berkel use Mobius concept for architectural design. Peter Eisenman was one of the first architects to use Mobius form to design “Max Reinhardt Haus” building (Fig. 5.11). He used Mobius strip as a volume with varied width and sliced it into many pieces and used it as an overall volume of building.

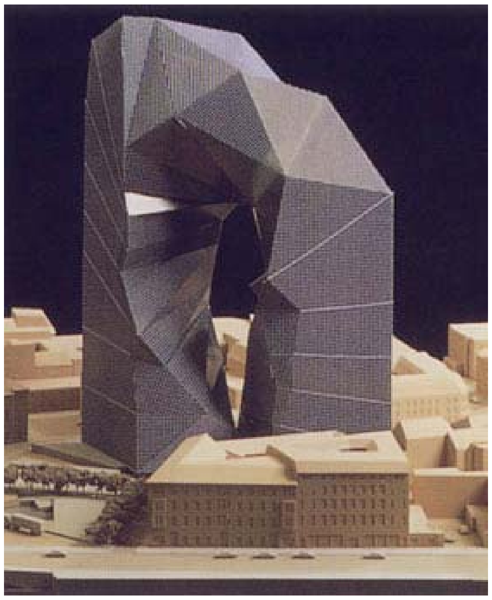


Fig. 5.11: Mobius building by Peter Eisenman.

- **Algorithmic Form Generation:**

Algorithmic form generation can be used to generate a variety of curves useful for generating architectural floor plans, architectural ornamentation as well as generating geometric spaces for building domes like Greg Lynn and Frank Gehry's organic forms and spaces.

Greg Lynn uses self intersecting algorithmic curves to generate volumetric pockets within continuous surfaces. According to Greg Lynn, blebs are pockets of space formed when a surface intersects itself, making a captured space. He uses a class of geometric curves beginning with the folium of Descartes and including Limacon of Pascal Maclaurin's Trisectrix, TschirnHaus's cubic, Cubic curves Freeth's, Nephroid Stronoid and Plateau curves (Fig. 5.12, 5.13, 5.14).

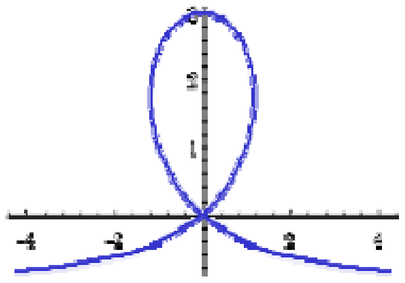


Fig. 5.12: Tri Of Maclaurin is used to create the volumes in figure 5.13.

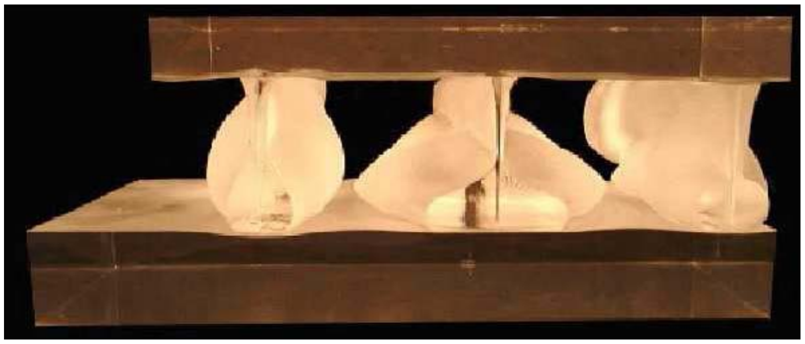


Fig. 5.13: St. Gallen Kunst Museum. Three volumes sandwiched between outdoor ceiling and St. Gallen Kunst Museum.

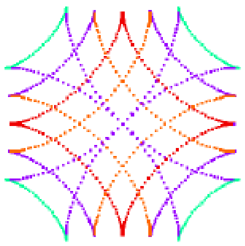
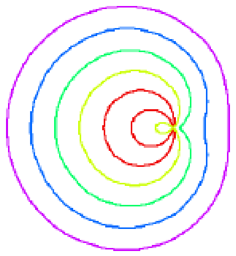



Fig. 5.14: Examples of astroid and limacon of pascal curves used by Greg Lynn for building design.

Mathematical algorithms may be used to generate other surfaces which are useful for architectural design. (Fig. 5.15, Fig. 5.16) illustrates Embryological House and Offices in New York, designed by Greg Lynn showing the use of algorithmic curves to form folding surfaces.

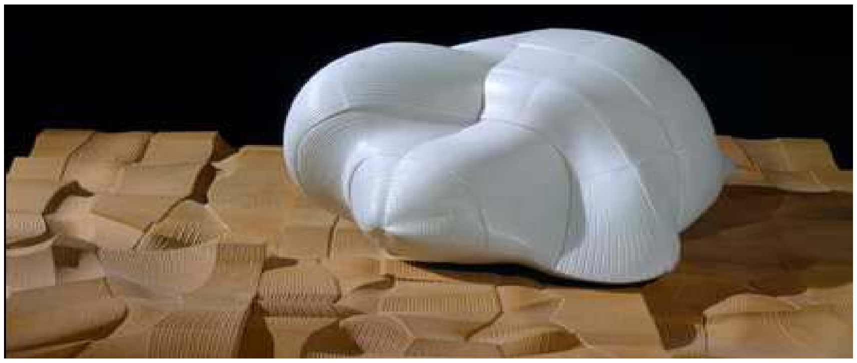


Fig. 5.15: Voluptuous undulating surfaces in Embryological House.

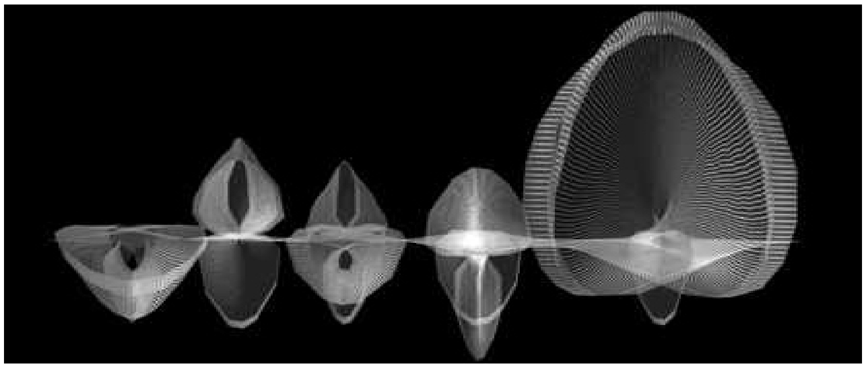


Fig. 5.16: Imaginary forces, New York offices by Greg Lynn.

- **Fractals:**

The IFS fractals (vector fractals), Complex number fractals (Point fractals) and the Orbit fractals (strange attractors) are the three types of fractals previously discussed in chapter three.

Peter Eisenman exhibited “House 11-a”, less than a year after the English language publication of Fractals: Form, Chance and Dimension by scientist Benoit Mandelbrot.

Peter Eisenman used the concept of fractal scaling in his house. House (11a) became a motif in Eisenman's housing. The process he describes philosophically as entailing "three destabilizing concepts: discontinuity, which confronts the metaphysics of presence; recursively, which confronts origin; and self-similarity, which confronts representation and aesthetic object".

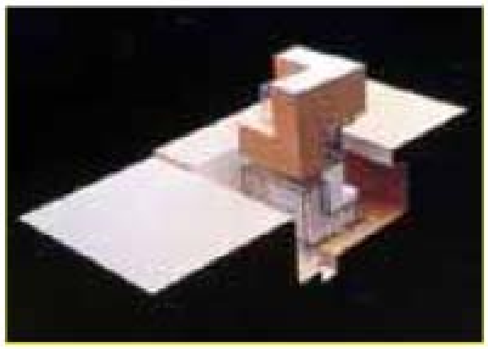
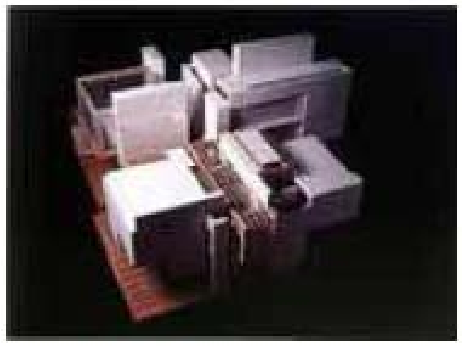



Fig. 5.17: Using fractal repetition in house 11-a and House X respectively.

House 11a and house X (Fig. 5.17), a composition of Eisenman’s signature “L”s is a combination of transformation rules; such as rotation and scaling. The “L” is actually a square which has been divided into four quarters and then one quarter square is removed. Eisenman viewed this resulting “L” shape as symbolizing an “unstable” or “inbetween” state; neither a rectangle nor a square. The three dimensional variation is a cubic octant removed from a cubic whole.

Point fractals as well as vector fractals can generate a wide variety of shapes and patterns that can be used in tiling, flooring, facades (Fig. 5.18), shaping plans and layouts (Fig. 5.19),. Heneghan Peng used Sierpinski's point fractal in the design of the translucent alabaster cladding of the front facade of the grand Egyptian museum.

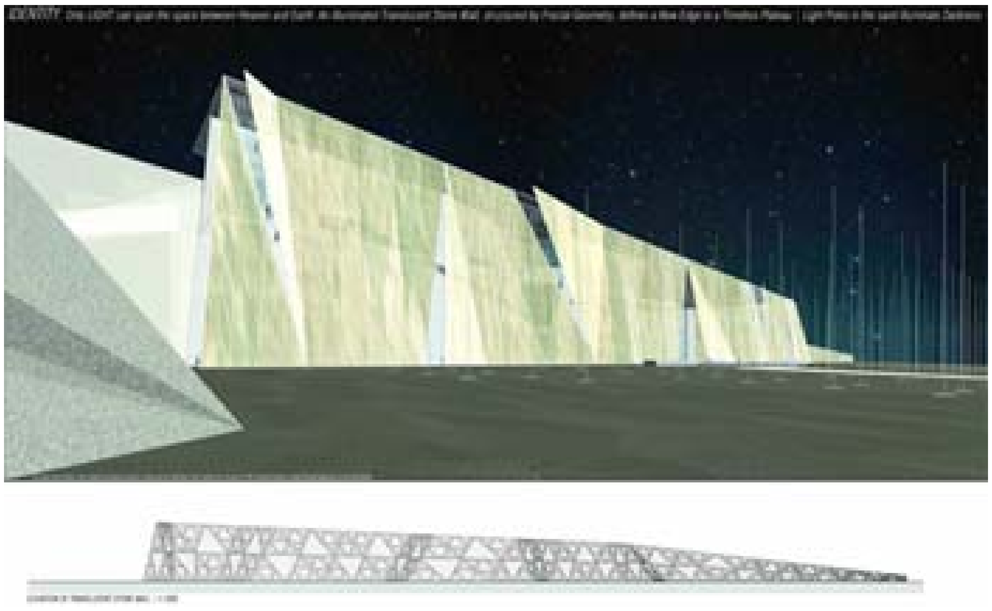


Fig. 5.18: Sierpinski set used in the translucent alabaster cladding of the grand Egyptian museum's façade.

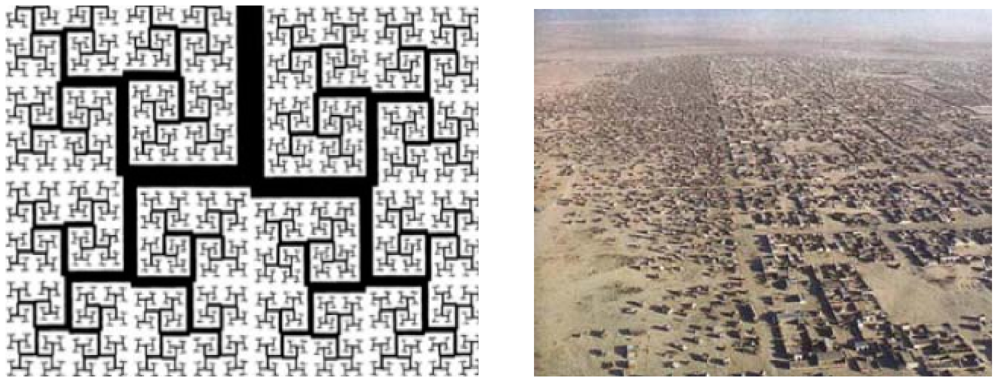


Fig. 5.19: H-fractal used in a housing layout in Nouakchott, Mauritania.

Orbit fractals (strange attractors) generate repeating point patterns in two-dimensional space while their coloring algorithms which represent time can produce images of coherent three-dimensional forms. The third dimension is determined by the perception of the viewer coupled with a created intent.

Endless numbers of three dimensional forms can be generated using a series of related strange attractor equations and polynomial functions. The product of these attractors may inspire broken forms and spaces resemble Naga's style (Fig. 5.20). As well as organic forms and spaces resembles Greg Lynn's style (Fig. 5.21).

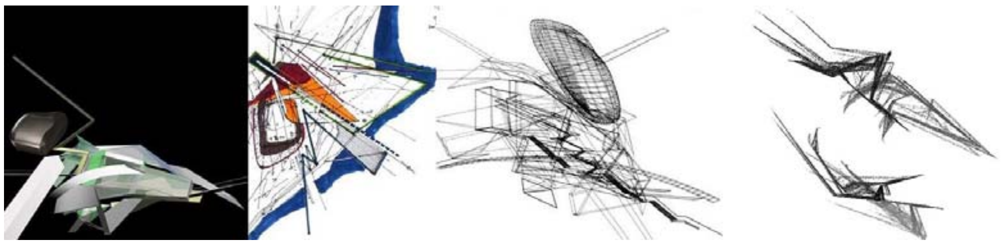


Fig. 5.20: Poly-function attractor (right) resembles sketches by architect T. Naga in Yokohama terminal (left).

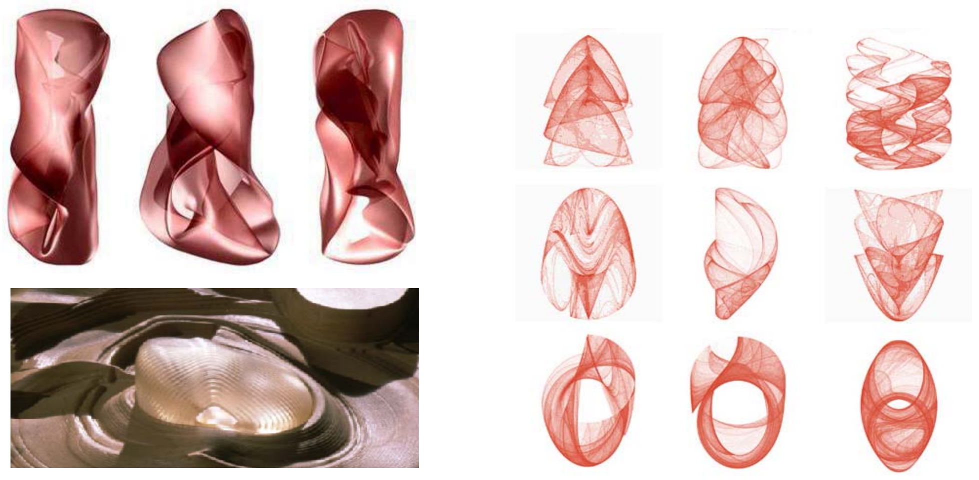


Fig. 5.21: Related attractors (right) resembles forms by T. Naga- Science Museum (left-up) and Greg Lynn- Embryological house (left-down).

- **Spirolaterals:**

Spirolaterals as a mathematical figure that generates novel geometric entities of unexpected complexity can be used for the purpose of inspiration and generation of two and three dimensional patterns and figures.

These geometric entities (Fig. 5.22) can be utilized in a number of methods to generate architectural forms. One is to technically layout architectural elements along these patterns as masses or master plans, another is to use these geometries in wall architecture, mass architecture, panel architecture, layered architecture, Infill architecture, skin architecture and landscape or patterns for doors and windows, tiling and ornaments (Fig. 5.23).

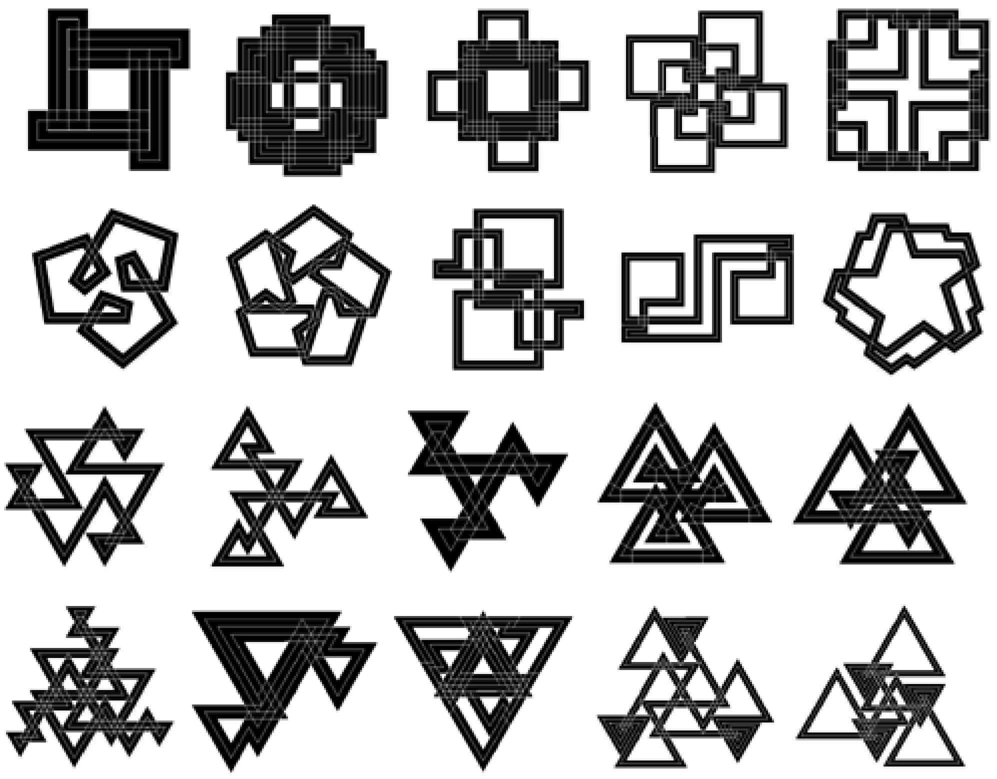


Fig. 5.22: A series of generated straight spirolaterals.

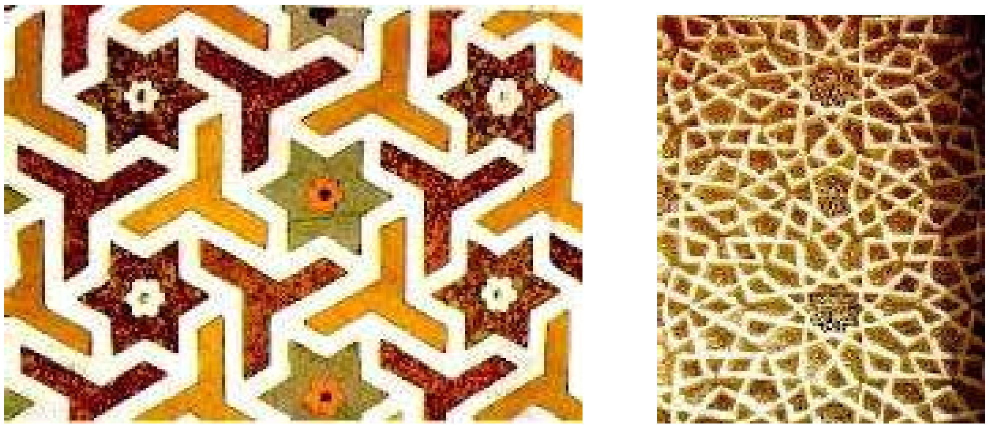


Fig. 5.23: Ornaments can be figured using generated spirolaterals.

Hypocycloid curves, epicycloids curves, antimercator, circular, normal and harmonic mean inversion are types of curved spirolaterals (Fig. 5.24) that empower design with curved lines.

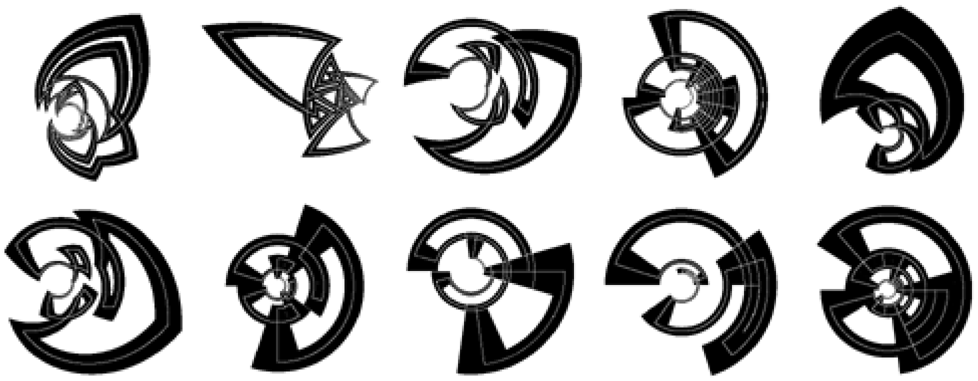


Fig. 5.24: A series of generated curved spirolaterals.

Barrionuevo and Borsetti (2001) extended the spirolateral definition to the three-dimensional space, introducing the concept of "Spirospaces". They developed a computer program to draw closed spiroraterals and they outlined the possibility to carry out rotations in three-dimensional space to generate more complex configurations (Fig. 5.25, 5.26, 5.27).

- Spirospace's geometry to architectural idea's form analogy. Spirospace element configuration inspires interior architectural solutions (Fig. 5.25).

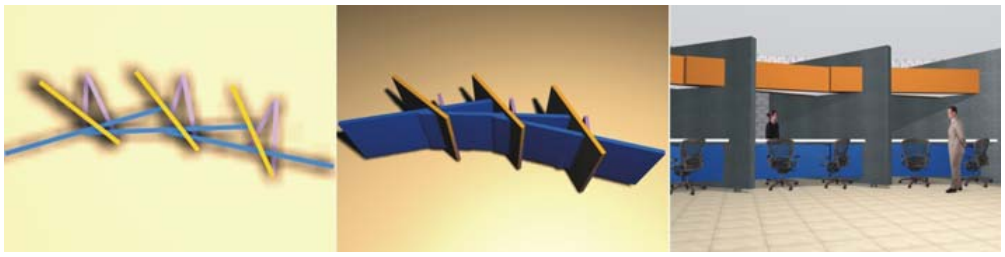


Fig. 5.25. Architectural interpretation of an interior space (R. Borsetti).

- Spirospace's syntax to architectural layout analogy (Fig. 5.26).

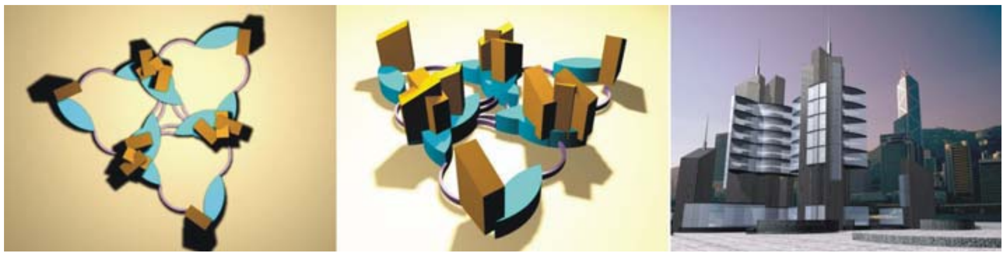


Fig. 5.26. Architectural layout from a spirospace interpretation (R. Borsetti).

- Spirospace's space to architectural function analogy (Fig. 5.27).

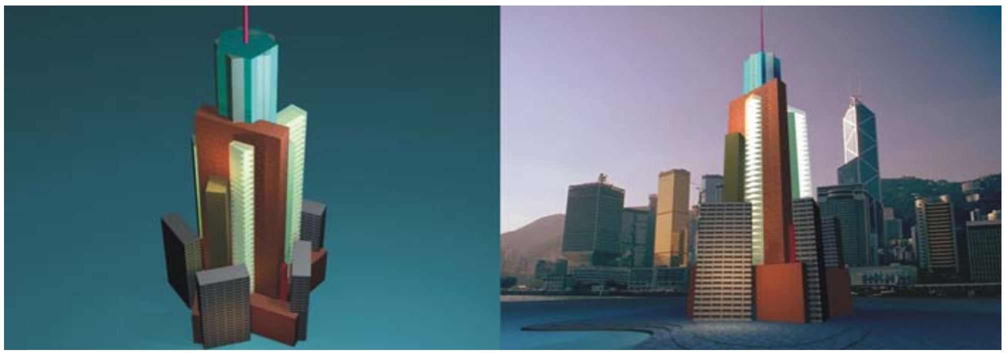


Fig. 5.27. Architectural interpretation of a skyscraper.

- **Genetic Algorithms:**

Using genetic algorithms in architecture models require architectural concepts to be described in the form of genetic codes. Then these codes are mutated and developed by computer programs into a series of populations. While models are evaluated by optimization or selection sub-systems, the codes of successful models are constantly picked up until a particular stage of development process is reached (Fig. 5.28).

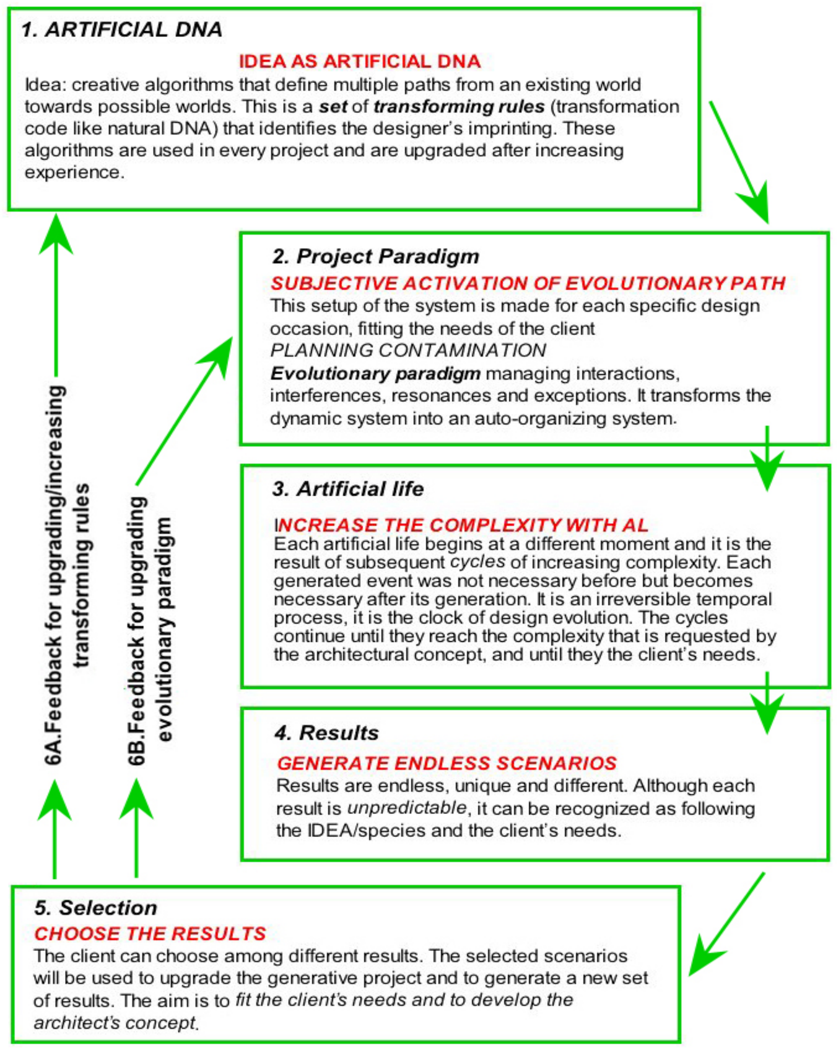

1. ARTIFICIAL DNA
IDEA AS ARTIFICIAL DNA
Idea: creative algorithms that define multiple paths from an existing world towards possible worlds. This is a set of transforming rules (transformation code like natural DNA) that identifies the designer's imprinting. These algorithms are used in every project and are upgraded after increasing experience.
2. Project Paradigm
SUBJECTIVE ACTIVATION OF EVOLUTIONARY PATH
This setup of the system is made for each specific design occasion, fitting the needs of the client
PLANNING CONTAMINATION
Evolutionary paradigm managing interactions, interferences, resonances and exceptions. It transforms the dynamic system into an auto-organizing system.
3. Artificial life
INCREASE THE COMPLEXITY WITH AL
Each artificial life begins at a different moment and it is the result of subsequent cycles of increasing complexity. Each generated event was not necessary before but becomes necessary after its generation. It is an irreversible temporal process, it is the clock of design evolution. The cycles continue until they reach the complexity that is requested by the architectural concept, and until they the client's needs.
4. Results
GENERATE ENDLESS SCENARIOS
Results are endless, unique and different. Although each result is unpredictable, it can be recognized as following the IDEA/species and the client's needs.
5. Selection
CHOOSE THE RESULTS
The client can choose among different results. The selected scenarios will be used to upgrade the generative project and to generate a new set of results. The aim is to fit the client's needs and to develop the architect's concept.
6A.Feedback for upgrading/increasing transforming rules
6B.Feedback for upgrading evolutionary paradigm

Fig. 5.28: The generative process cycle using GA.

Celestino Soddu, professor at Milan University, has been involved with creating many generative systems for various purposes based on genetic code scripts. He has written programs like Basilica, which creates endless sequences of architecture, and Argenia which is a design machine that makes endless generative forms (Fig. 5.29).

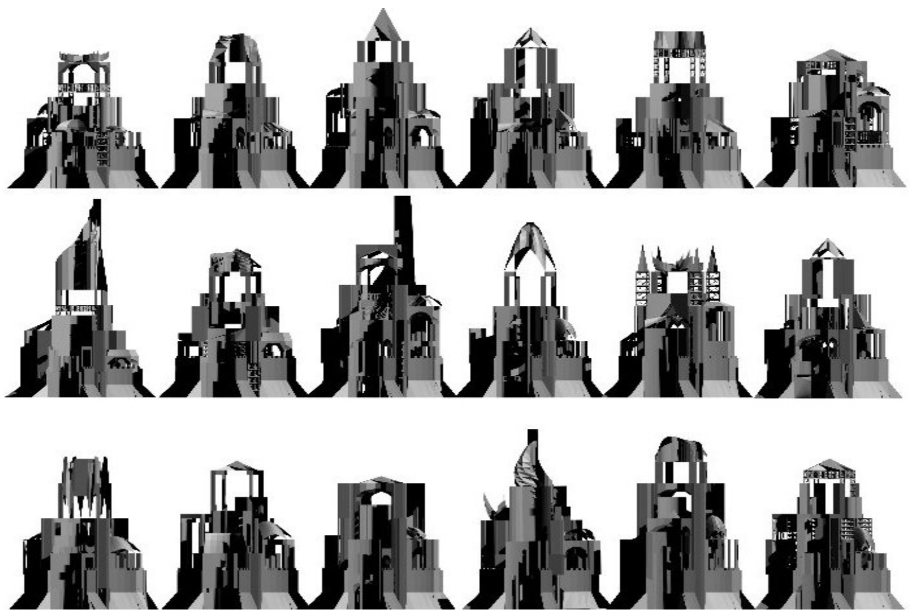


Fig. 5.29: Generated variations of Castles with Argenia by Soddu.

Since 1987, Celestino Soddu has been experimenting with creation of systems that can be used to generate unique objects, whether art, city planning, architecture, industrial design objects or graphic design.

The following images present a series of visionary scenarios of genetic applications in Hong Kong (Fig. 5.30), Washington DC (Fig. 5.31), New York (Fig. 5.32), Milan (Fig. 5.33 and 5.34) and Nagoya (fig 5.35). In these generative projects the architecture was designed and generated with Argenia, a genetic code script program, made by architect Celestino Soddu.

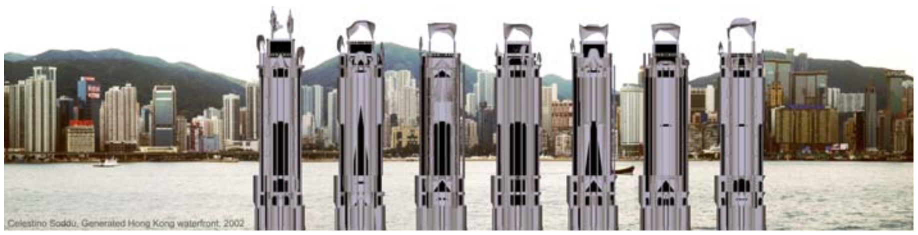


Fig. 5.30: Generated Variations of buildings for Hong Kong waterfront.

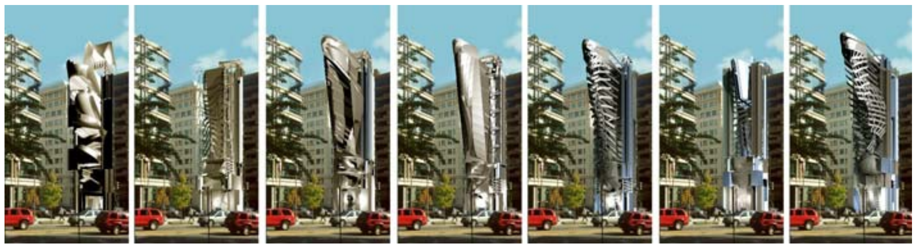


Fig. 5.31: Generated variations of IDB Cultural Centre, Washington DC.

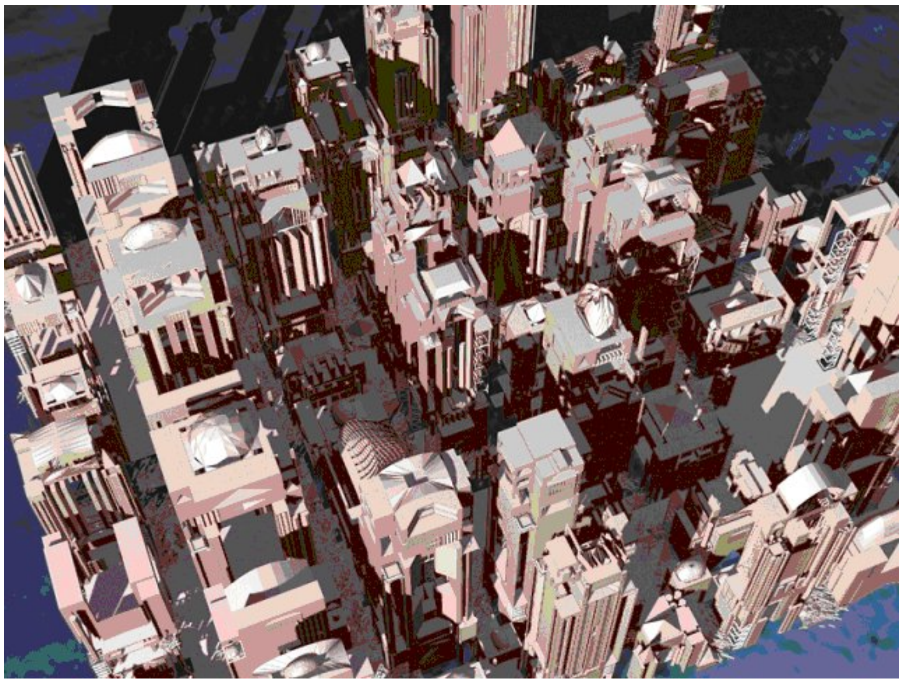


Fig. 5.32: Generated town system representing the New York identity code.

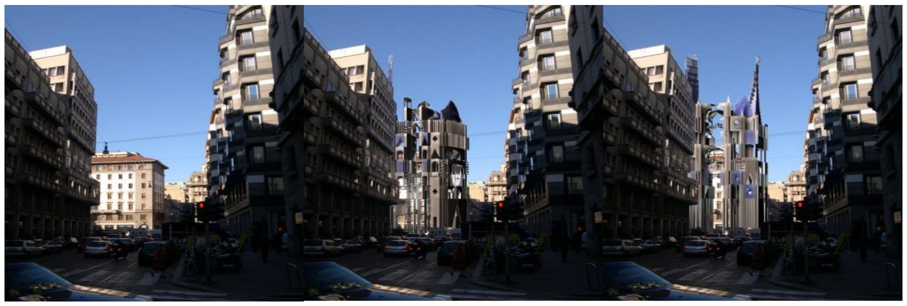


Fig. 5.33: 1 real + 2 generated Variation of a New Gallery in Milan.

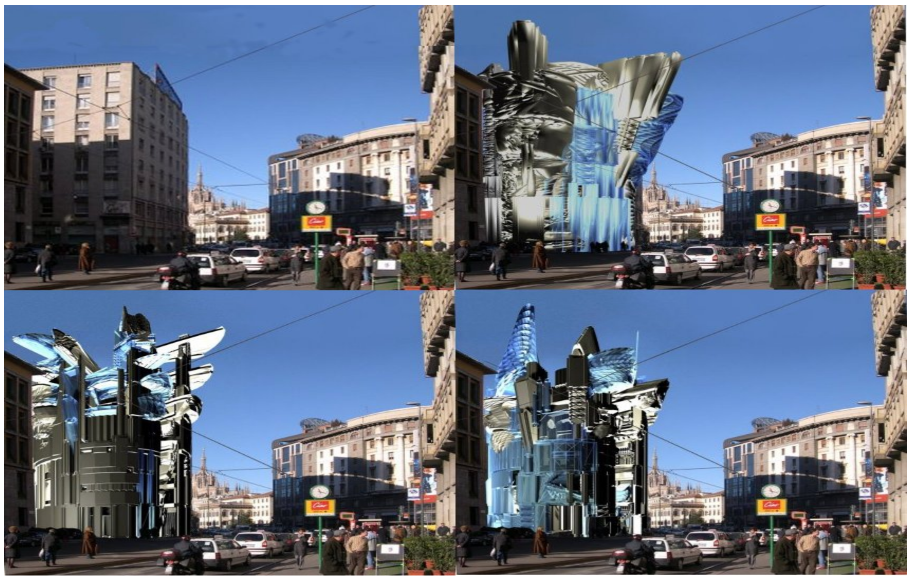


Fig. 5.34: 1 real+3 generated Variation of Milan New Museum of Futurism.

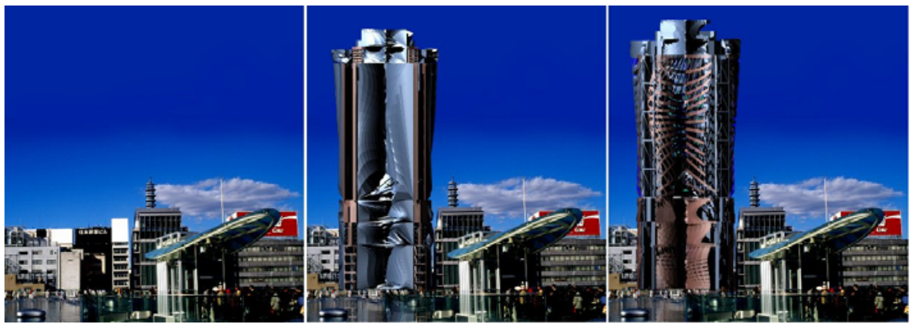


Fig. 5.35: 1 real + 2 generated Variation of a tower in Nagoya downtown.

- **Cellular Automata:**

The connection of Cellular automata to architecture is the ability to generate patterns. From these organized patterns it might be able to suggest architectural forms. Developments for the integration of CA into the architecture design process undergo in many academic researches. Some used CA as a recursive form generator in architecture, and other used it as generative strategy in architecture to generate varieties for high-density contexts.

Frazer, Coates, Watanabe, Krawczyk and Clarke, are architects undergoing researches on architectural interpretation of Cellular automata. Some of their works are shown in (Fig.5.36, Fig. 5.37).

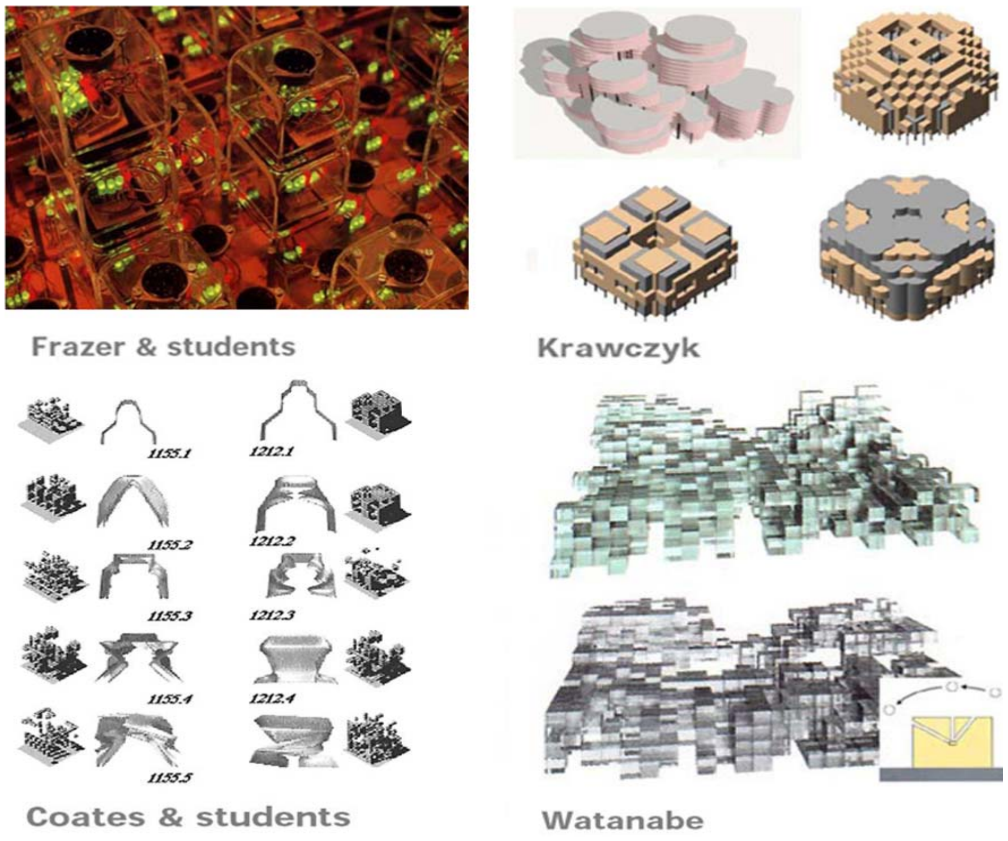


Fig. 5.36: Frazer, Coates, Watanabe, Krawczyk interpretations of cellular automata.

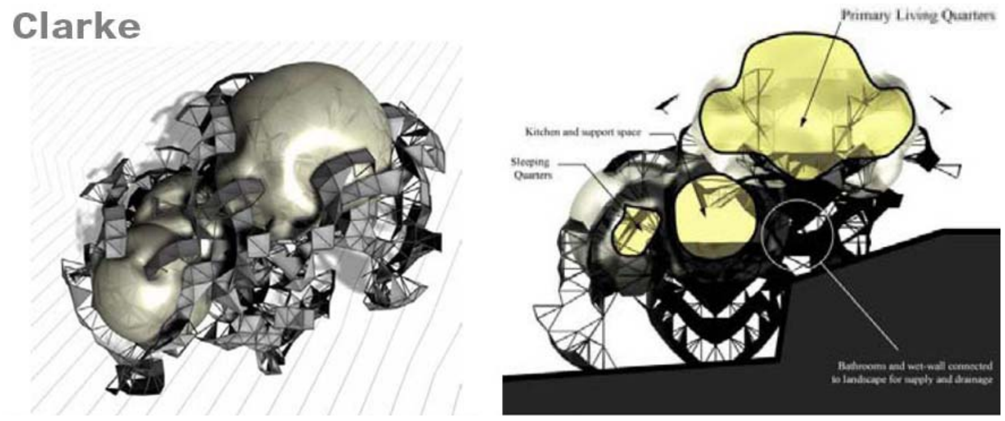

Clarke
Primary Living Quarters
Kitchen and support space
Sleeping Quarters
Bathrooms and wet-wall connected to landscape for supply and drainage

Fig. 5.37: Clarke interpretations of cellular automata using skinned envelopes.

Cero9 examined the generative design potential of cellular automata by applying them to the re-modeling of the northern style housing competition in Aomori/Japan 2001. (Fig. 5.38) shows the adaptation of CA in the design process. (Fig. 5.39, 5.40) shows the final results.

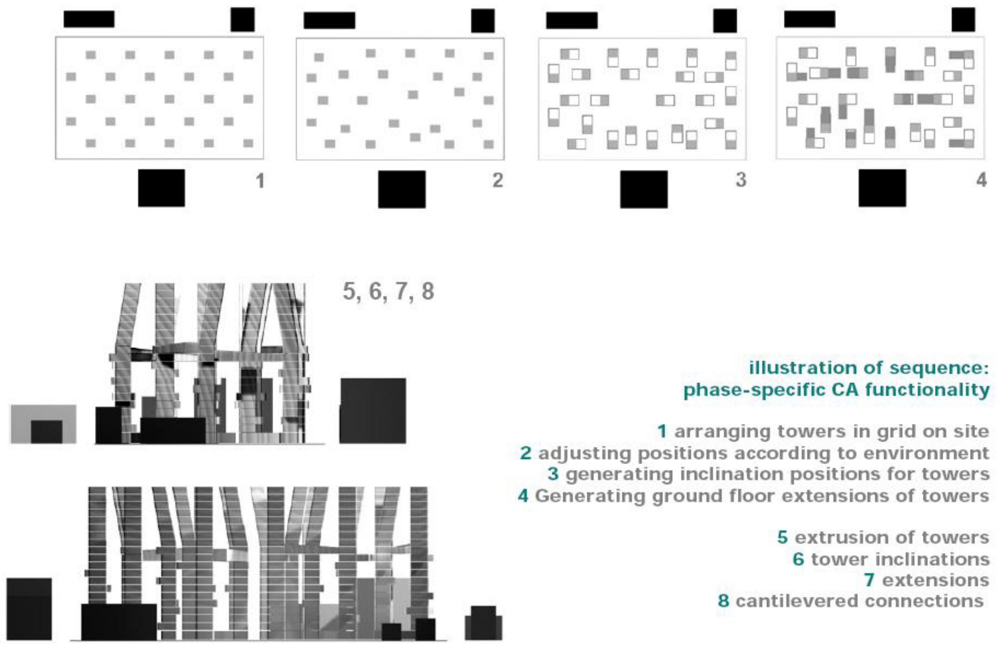

1
2
3
4
5, 6, 7, 8
illustration of sequence:
phase-specific CA functionality
1 arranging towers in grid on site
2 adjusting positions according to environment
3 generating inclination positions for towers
4 Generating ground floor extensions of towers
5 extrusion of towers
6 tower inclinations
7 extensions
8 cantilevered connections

Fig. 5.38: CA adaptation into the design process of the housing competition re-modeling.

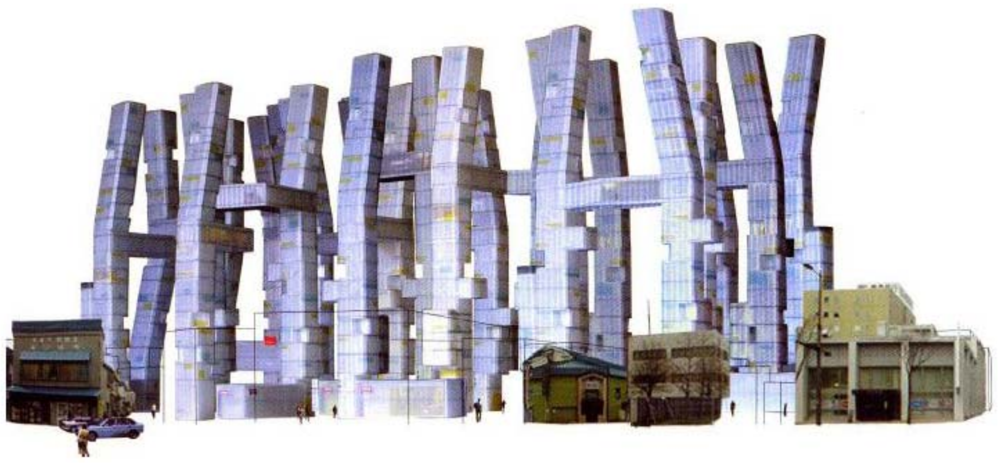


Fig. 5.39: The final outcome of the northern style housing competition re-modeling Aomori/Japan 2001 using CA.

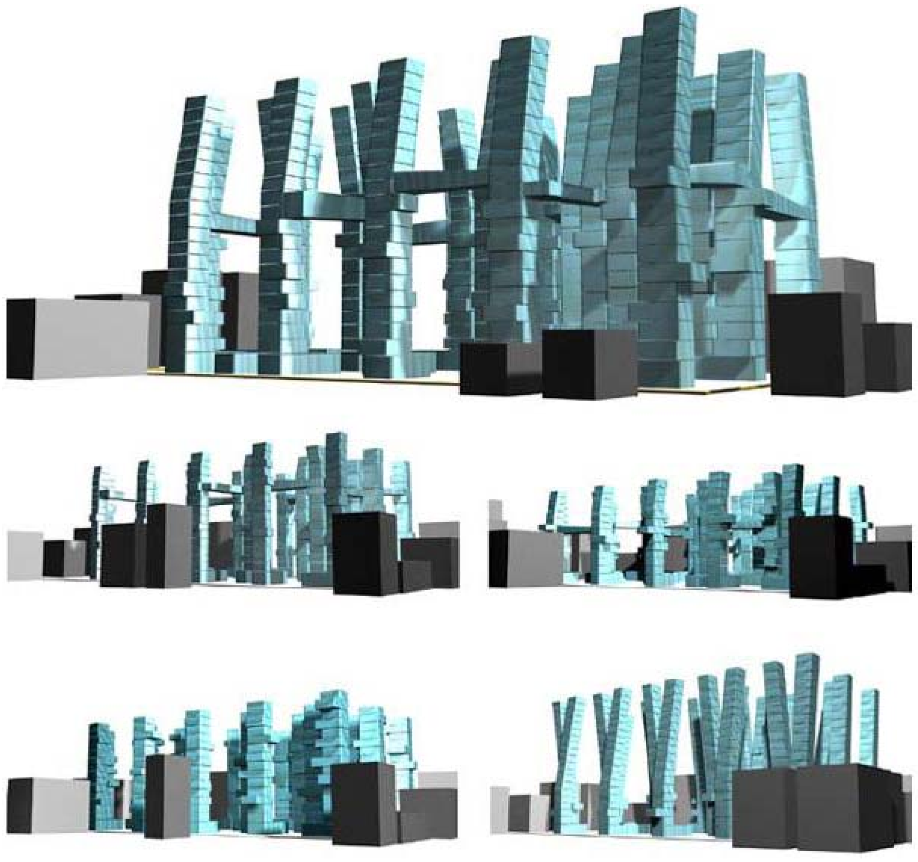
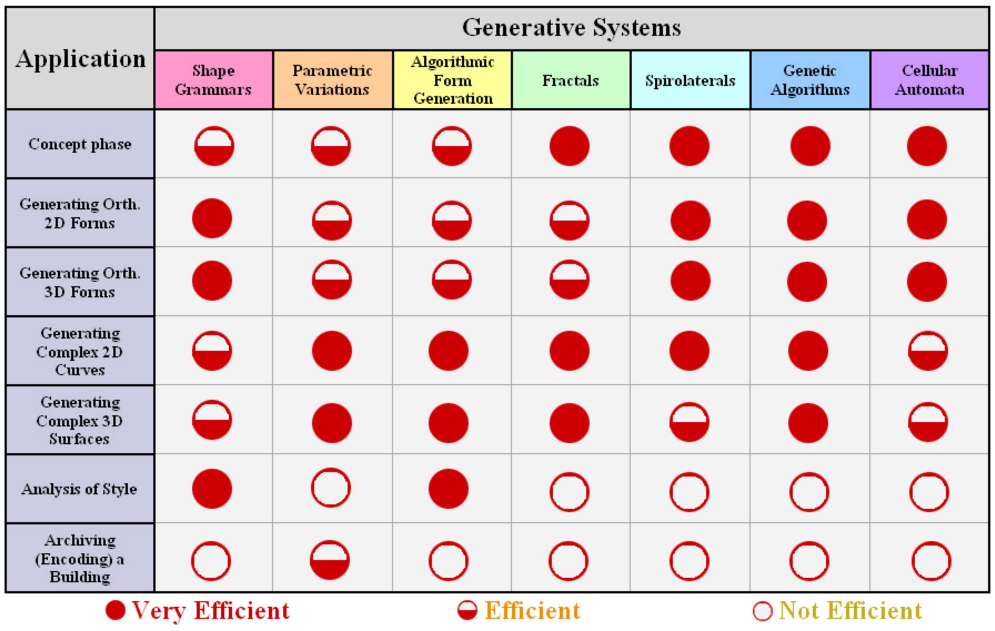


Fig. 5.40: Variations in outcome.

- Comparative Analysis

The following is an analytical comparative study for the efficiency of the use of each generative tool in different applications in the architectural design process (Tab. 5.1):



Application	Generative Systems						
	Shape Grammars	Parametric Variations	Algorithmic Form Generation	Fractals	Spirolaterals	Genetic Algorithms	Cellular Automata
Concept phase	◒	◒	◒	●	●	●	●
Generating Orth. 2D Forms	●	◒	◒	◒	●	●	●
Generating Orth. 3D Forms	●	◒	◒	◒	●	●	●
Generating Complex 2D Curves	◒	●	●	●	●	●	◒
Generating Complex 3D Surfaces	◒	●	●	●	◒	●	◒
Analysis of Style	●	○	●	○	○	○	○
Archiving (Encoding) a Building	○	◒	○	○	○	○	○

● Very Efficient ◒ Efficient ○ Not Efficient

Tab. 5.1: The efficiency of each generative tool in design.

Conclusions

- First, these geometric entities need to be guided, to be constrained, to be filtered and to be mutated by the utilitarian requirements of the functionalities of a building before being interpreted into any architectural element.

- Fractals, Spirolaterals, Genetic Algorithms and Cellular automata are very efficient for the concept phase in architectural design process.

- Shape grammar, Fractals and Spirolaterals can be utilized in a number of methods to generate architectural forms. They can relate to orthogonal architectural elements like axes or masses and zoning layout. Another method is to use these geometries in a variety of architectural elements like: wall architecture, panel architecture, skin architecture, landscape architecture or patterns for doors, windows, tiling and ornaments.

- Parametric Variations, Algorithmic Form Generation and Fractals (strange attractors in particular) can be used to generate a variety of complex curves and surfaces useful for generating organic designs, architectural ornamentations as well as generating geometric spaces for building domes like Greg Lynn's and Frank Gehry's organic forms and spaces.

- Fractals and Cellular Automata can be used to obtain high density housing layouts and urban planning.

- IFS Fractals can be used to analyze complex rhythms of facades of buildings (façade designs) and generation of patterns similar to Islamic architecture.

- Point Fractals are used in landscape and terrain generation and used in texture generation as well generation of clouds, mountains and plants.

- Spirolaterals can generate a great number of unexpected orthogonal or curved designs and Islamic patterns. The unpredictable under controlled conditions, is what makes the spirolateral of continuing interest. They could be small in scale, desktop size, or wall hung. They could be used in furniture, or in outdoors area for sitting, the assemblies could be scaled for a person to walk into. At smaller scale. Some could be created as jewelry.

- Genetic Algorithms and Cellular Automata show great power in design fields due to their ability of creating a wide range of alternatives in design in a very short time. The user acts as a judge driving selection using aesthetic judgments to breed art work.

5-2 Results

- The mathematical character of the generative tools could seem that their creation is something automatic, cold and distant from the sensibility and intimacy that is supposed to creativeness, but this is not true. The use of these tools to model objects simply constitutes a tool to work, like the hammer and chisel of the sculptor or the brush of the painter.

- Generative systems will not block human creative abilities. They are simply tools that allow the designer to explore more design possibilities and trigger inspiration and empower inventiveness.

- Changes in the architects' intuitive design methods should be accompanied by a shift in the mathematical basis of architectural design, as classical geometry can no longer be divorced from Algebra, Topology, Trigonometry, Chaos Mathematics and Evolutionary Algorithms.

- Functions of mathematical digital tools are no longer limited to two and three dimensional drafting and functional solutions or for final presentations as digital pens. They have become tools that can assist design thinking and form creation. And finally transformed into media generating unpredictable novel design.

- The computational tools currently used by the architects must evolve beyond their current limited representational use (e.g. CAD, 3D rendering) into a generative use.

- Generative art refers to any art where the artist uses a system, such as a set of natural language rules, a computer program, a machine, or other procedural invention, which is set with some degree of autonomy contributing to or resulting in a completed work of art.

- Generative systems are used for a variety of complex form design problems. They are especially useful in the concept generation phase of the design process where they are beneficial to consider a large number of possible solutions before proceeding to the selection and refinement phase.

- The difference between the conventional use of mathematics in design and the mathematical generative systems is that in the first the results can be easily anticipated, while in the second, the recursive manner of the mathematical functions of some generative systems makes the outcome usually can not be anticipated. This offers an interesting and rich platform from which to develop possible creative architectural forms.

- The process of generating architectural forms using mathematical constructions can be utilized in a number of methods. One is to technically layout architectural elements along such constructions, another is to explicitly develop forms corresponding exactly to the underlying concept, and another is to use such geometries as creative triggers and sources of inspiration for architectural design.

- The developments of generative design systems usually require programming skills, while their applications can be comparatively user-friendly and easy for non-programmers.

- Mathematical generative systems are powerful tools for creating design variance. But reducing design variance according to criteria of usefulness and beauty needs a great deal of knowledge and common sense. This common sense can not be put into software easily. Hence, in actual generative design projects, selections from design generations are typically performed by humans.

- Conventional CAD software is not easy to control in the design process, and it can be time consuming and labor-intensive to produce a complex shape configuration However, generative systems offer additional options that give Designers more complex form generating capabilities that quickly produce a large number of alternative designs, To be a design partner.

- Generative design is a scientific art process, thus the architect should be a mixture of artist/designer/programmer.

- The successful use of generative systems in other design field's conceptual design process, automotive design, aerospace design, furniture design and industrial design proves the feasibility of their use at an early stage of architectural design.

- Mathematical models can be abstracted into typological models. From this perspective, architectural objects are considered particular instances of typological models. This is an example of what, in academic communities, is called "to go from the form to the purpose". One of the ways to accomplish this is by using generative systems.

- The nonlinearity of chaotic systems results in the amplification of small differences. Chaotic system may seem random because its behavior is so unpredictable. But they are not random systems. Chaotic systems may be difficult to predict but they will still exhibit structure that is different than purely random systems.

- Evolutionary design is used in architectural design and in many engineering fields, to improve a previous design, or to create new design from scratch. Though successfully applied in other fields of engineering, still waits to be applied broadly in architectural design.

- Generative systems free the designer to focus more on choice, evaluation and refinement rather than generation which is entirely left to the autonomy of the computer.

- The evolutionary mathematical structures of these generative tools are capable of creating music as well as forms.

- Generative art is neutral. It is neither modern nor post modern. It is simply a way of creating art and any content considerations are up to the given artist. Certainly one can make generative art that exhibits a postmodern attitude. And others can make generative art that attempts to refute post-modernism.

5-3 Recommendations

With respect to architects:

- If the designer does not master the internal working of these new tools thoroughly, he can neither develop nor express his creativity. Therefore the real support of the form finding process requires one to learn the new methods of using computers and the evolved generative approach, particularly in the early stages of the process.

- To remain relevant in the rapidly changing design field the architects must maximize their use of the latest mathematical design tools during conceptual design.

- One can not master the use of the generative systems without a thorough understanding of the mathematical principles involved. Therefore, in design courses, computer-based application and creativity should be supported by "mathematics for design" courses.

With respect to academic teaching:

- Growing recognition of the importance of generative design methodologies must be a priority and theories and applications of generative design must be introduced to undergraduate students as part of their design studies.

- Teaching generative techniques initially requires the introduction of generative toolbox. This toolbox contains mathematical techniques, which in early teaching stages should be introduced in breadth rather than in depth.

- Architectural computing research must shift a bit from fundamental CAD developments towards higher applications of CAD tools. And computer programming should be taught within the curriculum in architectural schools.

- Generative design oriented textbooks are lacking at the moment, for that introductory materials must be developed and published.

The developments of advanced computer techniques and mathematical systems surely influences architecture and that is almost inevitable. We should not be afraid that these techniques and systems will take our place, because humans can only have wisdom. Though, some architects already decided to run parallel with technology, because they have realized that if they would ignore these developments, the technological culture might just go on without them, and no one likes the feeling of being deserted and left behind.

Summary of chapter five:

Chapter five covered a deduction for the characteristics of these generative systems with listing some applications for these generative tools in architecture and an analytical comparison study for the efficiency of each of generative tool in different phases in architectural design process. And finally, results and recommendations are presented at the end of the chapter.

REFERENCES

REFERENCES

Abelson, H. and Disessa, A.: *Turtle Geometry*. MIT Press, pp.37-39, 120-122, 1968.

Abimbola, O. A.: *Exploring Algorithms as Form Determinants in Design*. The University of Oklahoma, USA, The 3rd International Space Syntax Symposium, Atlanta, 2001.

Barrallo, J.: *Geometria Fractal*. Madrid, Anaya Multimedia, 1992.

Bentley, P.: *Evolutionary Design by Computers*. Morgan Kaufmann publishers, 1999.

Boesiger, W.: *Le Corbusier*. Ingoprint, Barcelona, 1992.

Bourke, P.: *The Pattern Book: Fractals, Art and Nature*. Edited by Pickover C., World Scientific Publishing, Singapore, 1995.

Bovill, C.: *Fractal Geometry in Architecture and Design*. Birkhauser, Boston 1996.

Chapuis, J. and Lutton, E.: *ArtiE-fract: Interactive Evolution of Fractals*. International journal on artificial intelligence tools, 15 December, 2005.

Chiarella, M.: *Geometry and Architecture: NURBS, Design and Construction*. Proceedings of the Fourth International Conference of Mathematics and Design, special edition of the journal of Mathematics & Design, volume 4, no.1, pp.135-139, 2004.

Ching, F. D. K.: _Architecture Form, Space and Order_. Van Nostrand Riendold, 1979.

Chinowsky, P. S.: _The CADDIE Project: Applying Knowledge-Based Paradigms to Architectural Layout Generation_. Ph.D thesis department of civil engineering, Stanford University, May 1991.

Coates, P.: _The Use of Cellular Automata to Explore Bottom Up Architectonic Rules_. Eurographics Conference, Imperial College of Science and Technology, London, 1996.

Couceiro, M.: _Architecture and Biological Analogies, Finding a Generative Design Process Based on Biological Rules_. Escola Tecnica Superior d´Arquitectura – Universidad Internacional de Catalunya, Spain, eCAADe 23 - session 13: generative systems, 2005.

Cromwell, P. R.: _Polyhedra._ Cambridge University Press, 1997.

Devaney, R.: _Chaos and Fractals, The Mathematics Behind the Computer Graphics_. American Mathematical Society, Providence, p.141. 1989.

Dixon, R.: _Mathographics_. Dover Publications, Inc., p.152, 1987.

Dürr, C.: _Morphogenesis, Evolution of Shape_. ETHZ, CAAD, NDS 2004.

Eisenman, P.: _Peter Eisenman Diagram Diaries_. Universe Publishing, New York, p. 27-43, 1999.

Eldaly, H.: _Architecture in The Age of Information Technology._ M.Sc. thesis in architecture, Ain Shams University, 2005.

Elezkurtaj, T. and Franck, G.: *Genetic Algorithms in Support of Creative Architectural Design*. Vienna University of Technology, Department of Computer Aided Planning and Architecture, Vienna, Austria, eCAADe conference proceedings, Liverpool (UK) 15-17 September 1999.

Flemming, U.: *Syntactic Structures in Architecture*. M. McCullough, W. J. Mitchell, and P. Purcell, eds., The Electronic Design Studio, The MIT Press, Cambridge, pp. 31-47, 1990.

Fowler, D.: *The Mathematics of Plato's' Academy*. Clarendon Press, Oxford, 1999.

Frazer J., Frazer, J., Liu, XY., Tang, MX. and Janssen, P.: *Generative and Evolutionary Techniques for Building Envelope Design*. GA2002 (Generative Art and Design Conference, Politecnico di Milano University, Italy, Milan 11-12-13 December 2002).

Garcia, F., Fernandez, A. and Barrallo, J.: *Discovering Fractal Geometry in CAAD*. eCAAD 1994, Proceedings (conversion 2000).

Gardner, M.: *The Fantastic Combinations of John Conway's New Solitaire Game of "Life*". Scientific American, 223, pp. 120-123, 1970.

Gardner, M.: *Knotted Doughnuts and Other Mathematical Entertainments*. W. H. Freemand and Company, pp. 205-208, 1986.

Gero, J. S. and Maher, M. L.: *Mutation and Analogy to Support Creativity in Computer Aided Design.* Design Computing Unit University of Sydney, NSW, 2006, Australia, CAAD futures Digital Proceedings 1991

Gross, M. D.: *FormWriter: A Little Programming Language for Generating Three-Dimensional Form Algorithmically.* Computer Aided Architectural Design Futures 2001, Kluwer Academic Publishers, 2001.

Goldberg, D. E.: *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

Heitor, T., Duarte, J. P. and Pinto, R. M.: *Combining grammars and Space Syntax: Formulating, Evaluating and Generating Designs*. The 4th International Space Syntax Symposium, London, 2003.

Holland, J. H.: *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

http://mathworld.wolfram.com . Accessed 15/9/2007.

http://www.btinternet.com/~ndesprez/manual/attractors.htm. Accessed 23/10/2007.

http://www.generativeart.com. Accessed 15/7/2007.

Ibrahim, M. and Krawczyk, R. J.: *Exploring the Effect of Direction on Vector-Based Fractals*. College of Architecture, Illinois Institute of Technology, USA, Bridges 2002 conference, Towson, MD, July, 2002.

Ibrahim, M. and Krawczyk, R. J.: *Generating Fractals Based on Spatial Organizations*. College of Architecture, Illinois Institute of Technology, USA, February 2001.

Jadwiga, C. Z.: *Chaos, Databases and Fractal Dimension of Regional Architecture*. PhD in Architecture, Faculty of Architecture TU of Bialystok, Poland, eCAADe conference proceedings, Paris, France, 24-26 September 1998.

Jakimowicz, A., Barrallo, J. and Guedes, E. M.: *Spatial Computer Abstraction: From Intuition to Genetic Algorithms*. CAAD futures Digital Proceedings, 1997.

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

Kalay, E. Y.: *Modeling Objects and Environment.* John Wiley & sons, 1987.

Knight, T. W.: *Shape Grammars in Education and Practice: History and Prospect*. Online paper, Department of Architecture, MIT, 2000.

Kolarevic, B.: *Digital Morphogenesis.* In B. Kolarevic, (ed) Architecture in the Digital Age, Design and Manufacturing. New York: Spon Press, 2003.

Koning, H. and Eisenburg, J.: *The Language of the Prairie: Frank Lloyd Wright's Prairie Houses*. Environment and Planning B8, 295-323, 1981.

Krawczyk, R. J.: *www.netcom.com/~bitart.*2002. Accessed 1/10/2007.

Krawczyk, R. J.: *Architectural Interpretation of Cellular Automata*. Illinois Institute of Technology, USA, Generative Art 2002.

Krawczyk, R. J.: *Dimension of Time in Strange Attractors*. College of Architecture, Illinois Institute of Technology, USA. ISAMA, Bridges Conference, 2003.

Krawczyk, R. J.: *Experiments in Architectural Form generation Using Cellular Automata*. Illinois Institute of Technology, College of Architecture, USA, (design e-education) Modeling Real and Virtual Worlds Session 15, eCAADe 2002.

Krawczyk, R. J.: *More Curved Spirolaterals*. College of Architecture, Illinois Institute of Technology, USA. BRIDGES: Mathematical Connections In Art, Music, and Science, 2001.

Krawczyk, R. J.: *Sculptural Interpretation of a Mathematical Form*. College of Architecture, Illinois Institute of Technology, USA, Bridges, Towson University, 2002.

Krawczyk, R. J.: *Spirolaterals, Complexity from Simplicity*. In International Society of Arts, Mathematics and Architecture, 1999.

Krawczyk, R. J.: *The Art of Spirolateral Reversal*. The International Society of Arts, Mathematics and Architecture, edited by N. Friedman, University of Albany-SUNY, 2000.

Kwon, D. Y.: *ArchiDNA: A Generative System for Shape Configuration*. Master of Science in Architecture, University of Washington, USA, 2003.

Kwon, D. Y., Gross, M. D. and Yi-Luen Do, E.: _ARCHIDNA :A Generative System for Shape Configuration_. Design Machine Group, University of Washington, USA, 2003.

Lawrence, J. D.: _A Catalog of Special Curves_. Dover Publications, Inc., p.43, 1972.

Levin, P. H.: _Use of Graphs to Decide the Optimum Layout of Buildings_. Architect, 14, p.p 809–815, 1964.

Loomis, B.: A _Note on Generative Design Techniques: SGGA a User-Driven Genetic Algorithm for Evolving Non-Deterministic Shape Grammars_. Massachusetts Institute of Technology, Cambridge MA, USA, 2000.

Luis, F. B., Roberto, G. L. and Roberto, S.: _Spirospaces in Architectural Design_. Design Systems Laboratory, University of Tucuman, Argentina, 1st ASCAAD International Conference, e-Design in Architecture KFUPM, Dhahran, Saudi Arabia. December 2004.

Maeda, J.: _Design by Number_. The MIT Press. Cambridge, MA, 1999.

McCormack, J., Dorin, A. and Innocent, T.: *Generative Design: A Paradigm For Design Research*. In Redmond, J. et. al. (Eds) Proceedings of Futureground, Design Research Society, Melbourne. (2004).

Manoa, D.: _Computer Generated Complex Curved Surfaces as an Architectural Design Tool_. A paper presented to the Third International Symposium on Asia Pacific Architecture, 1999.

McGill, M.: *A Visual Approach for Exploring Computational Design.* SMArchS Thesis, Department of Architecture, Cambridge, MA: Massachusetts Institute of Technology, 2001.

Michele, E.: *Mathland, from Flatland to Hypersurfaces*. Birkhauser-Publishers for Architecture, 2004.

Mitchell, W. J.: *The Logic of Architecture*. The MIT Press. Cambridge, MA, 1994.

Mohammadi, G.: *Geometric Shape Generator*. M.Sc. thesis proposal, University of Washington, 2004.

Mortenson, M. E.: *Geometric Modeling*. John Wiley and sons, 1985.

Nophaket, N.: *The Graph Geometry for Architectural Planning*. Journal of Asian Architecture and Building Engineering, May 2004.

Odds, F.: *Spirolaterals*. Mathematics Teacher, pp.121-124, 1973.

ONeill, B.: *Elementary Differential Geometry*. New York: Academic Press. 1966.

Pentilla, H.: *Describing the Changes in Architectural Information Technology to Understand Design Complexity and Free-Form Architectural Expression*. Helsinki University of Technology HUT, Department of Architecture, Finland, 2006.

Pickover, C.: *Chaos in Wonderland*. St.Martin's Press, New York, 1994.

Plato: Timaeus. In the Great Books of the Western World. Encyclopedia Britannica, London, Vol. 7, Plato, p. 442-477, 1052.

Prousalidou, E.: A Parametric System of Representation Based on Ruled Surfaces. Master of Science in Adaptive Architecture & Computation, Bartlett School of Graduate Studies. University College London, September 2006.

Rosenman, M. A. and Gero, J. S.: Evolving designs by generating useful complex gene structures. In P. Bentley (ed.), Evolutionary Design by Computers, Morgan Kaufmann, London, pp. 345-364. 1999.

Schrandt, R. and Ulam, S.: On Recursively Defined Geometrical Objects and Patterns of Growth. In A. Burks (Ed), Essays on Cellular Automata, University of Illinois Press, Urbana, pp. 232-243, 1970.

Shakiban, C. and Berstedt, J. E.: Generalized Koch Snowflakes. In Bridges: Mathematical Connections in Art, Music and Science, 1998.

Soddu, C.: Generative Natural Flux. Proceedings of Generative Art Conference, Milan, AleaDesign Publisher, December 2001.

Stiny, G. and Mitchell W. J.: The Palladian Grammar. Environment and Planning B5, no.1: 5-18, 1978.

Stiny, G.: Computing with Form and Meaning in Architecture. Journal of Architectural Education, 39(1): 7-19, 1985.

Stiny, G.: *Ice-ray: A Note on Chinese Lattice Designs*. Environment and Planning B 4: 89-98, 1977.

Tapia, M. A.: *GEdit, A Visual Implementation of a Shape Grammar System*. Environment and Planning B: Planning and Design, vol. 26, pp. 59-73, 1999.

Todd, S. and Latham, W.: *Evolutionary Art and Computers.* Academic Press, 1992.

Von Neumann, J.: *The General and Logical Theory of Automata*. In J. von Neumann, Collected Works, edited by A. H. Taub, 1963.

Wang, Y.: *3D Shaper, 3D Architecture Form Synthesizer*. SMArchS Paper, Department of Architecture, Cambridge, MA: Massachusetts Institute of Technology, 1999.

Weisstein, E. W.: *Concise Encyclopedia of Mathematics*. CD-ROM edition 1.0, Chapman & Hall/CRCnetBASE, 1999.

Wolfram, S.: A *New Kind of Science*. Wolfram Media Press, Champaign, 2002.

Yang, D. and Tang, M.: *Genetic Evolution: A Synthetic Approach in Form Generation.* Savannah College of Art and Design(SCAD),Georgia,USA,http://genetic.ming3d.com/GE_FEIDAD3.pdf. Accessed 16/10/2007

Yessios, C. I.: *A Fractal Studio*. In ACADIA '87 Workshop Proceedings, 1987.