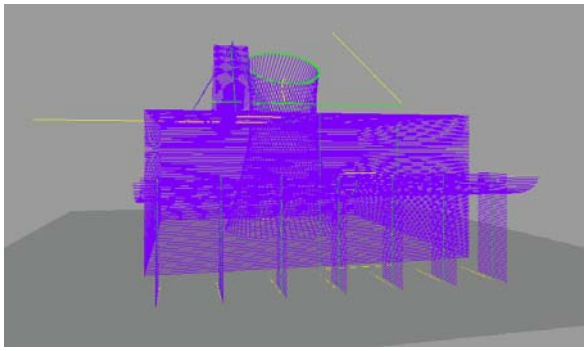
A Parametric System of Representation based on Ruled Surfaces

Elena Prousalidou



This dissertation is submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Adaptive Architecture & Computation
from the University of London

Bartlett School of Graduate Studies
University College London

September 2006

Abstract

The emergence of digital architectures reveals a gradual transformation of the design philosophy and consequently the design process itself. This thesis intends to explore the potential held by a representation of architectural forms that departs from the conventional methods of representation; a representation that is adjusted to the digital design framework.

The thesis hypothesizes that it is possible to create a simple parametric system to generate an almost complete set of building types based on ruled surfaces.

The aim is to investigate whether a parametric description of architecture, which fully exploits the power of information technology, can sufficiently represent architectural forms and whether it is able to enhance the design process. The possibility of an extended, more "active" role representation may play in the design process is also considered.

Various approaches that use a numerical description of form are discussed and some special features of Parametric Design are highlighted in order to draw up a list of criteria for the development of the parametric system. Its efficiency is tested by constructing a program and implementing it on a series of existing structures. The output is evaluated by the amount of information the system can provide and by its comparison to other methods of representation.

The thesis suggests that the parametric system that has been developed can be quite successful. The possibilities afforded by the system are discussed as well as its current limitations. Finally suggestions are made for further improvement.

Words: 9.218

Table of contents

List of illustrations

Acknowledgements

With very many thanks to:

My supervisors

Sean Hanna for his meticulous supervision, inspirational discussions and encouragement

Chiron Mottram for his instant interest and valuable advice

My friends

Eva Friedrich and Emily Chang for their input that helped me channel my ideas

Monika Brzozowski and Qandeel Shaam for supporting me

1. Introduction

"Parametric design allows the designer to treat a design as one large database adventure where design process decisions are published as histories embedded in the representation of the design in any given instance of its development." (Burry 2003)

Until recently, the integration of computers in architectural practice took the form of using computer-aided design tools as sophisticated drafting tools or tools to present architectural projects. Now, new digital techniques have emerged that attempt to exploit the powerful means provided by computers as a direct tool to generate novel architectural forms. Demonstrative examples of this new field of design techniques are Parametric Design – optimization- and Evolutionary Design -generative- approaches. Their common feature is the design of the end product as an instance of a class of possible solutions. Another similarity they share is the description of the product by a series of parameters that take on different values. These values can either be input by the designer, in the first case, or be evolved by a program, in the latter.

Seeing these methods simply as tools to enhance the design process would be an oversimplification. What lies underneath is a gradual transformation of the design philosophy that lines up with the overall changes in every aspect of contemporary activity brought by the digital revolution. Conventional architectural models are no longer sufficient to reflect modern life and deal with the problems of increasing complexity. Innovative design methods are required so that architecture becomes adjusted to the new cultural and technological conditions.

The new design framework that steadily develops includes representations. As stated by Branco Kolarevic, "predictable

relationships between the design and representations are abandoned in favour of computationally generated complexities" (Kolarevic 2003). This can be interpreted as a growing need for a different approach to representation of architectural form, interdependent to new design processes.

Attempts to represent architecture in a controversial way inspired by computation can be traced back to the 1970's when Lionel March developed a system to encode built form as a series of binary digits. Since then many systems of representation have been developed, each with different motivation and aims. A system of form description doesn't have to be unique or generalised; it can probably work better if targeted at a specific stage or area of design. A variety of methods is desirable. In the area of evolutionary design, for example, representations tend to be almost problem-specific.

There are some factors, though, that should be taken into account. It has become clear that, although the greater part of the existing built environment is constituted of rectangular forms and current constructional activity depends largely on planar forms, architectural interest is shifting towards curvilinear, more complex forms. The fascination with curves and complex surfaces is supported by new manufacturing techniques that allows the construction of irregular forms. Regarding representation, this indicates that a system can be efficient only if it represents curved as well as planar surfaces. Moreover, considering the constructional aspect of the built form in its representation is important as it allows for a better awareness of the end product.

"Generic representations form the vehicle for knowledge acquisition in a wide array of sources of architectural design knowledge."
(Achten, Oxman and Bax 2002)

2. Background

Architectural structures as numerical systems

Early attempts to give mathematical expression to shape were directed at a variety of form representations, such as modular and rectangular, non-modular rectangular spaces or irregular polygonal spaces (March 1972).

A typical early method of describing floor plans (March and Steadman 1971), was the description of irregular n-gons by listing the vectors of their vertices in cyclic order in a 2xn matrix. In the set-theoretical approach, rectangular forms were defined by component sets using Cartesian coordinates and distances in a 3x2 matrix, and the binary operations (∪) and (∩)were used for the union and intersection of sets of points.

Most of the approaches to three-dimensional form representation were based on a modular 'building block' description, having shapes formed by three-dimensional cubelets.

The method was probably first applied for architectural purposes in the 1930's, when Albert Fthearwell Bemis proposed that a cube might be used as a module in building design and component standardization. The building could be designed within a 'total matrix of cubes'. Subtraction of cubelets from the volume would result in the definition of exterior surface, followed by the elimination of volumes within the house.

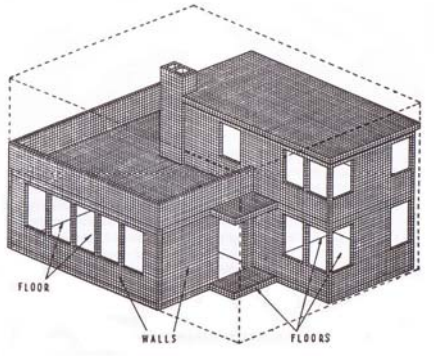


Figure 1. 'The Evolving House' House structure defined within a matrix of cubelets. (March and Steadman 1971, p200)

With his isomorphic[1] Boolean approach March (March, 1972) proposes the description of a rectangular object using binary encoding. The method makes use of cells and the set of points generated by the product of the two dimensioning sets X1 and X2. Each cell is uniquely identified by the coordinates of one of its vertices. The ordered cells can be unfolded into a sequence of atoms that form a chain.

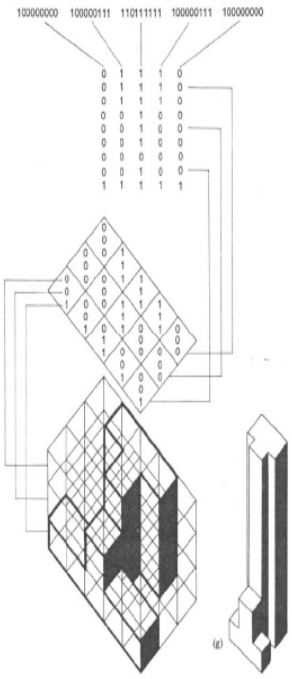


Figure 2. Encoding the Seagram building. (March 1972)

He illustrates his theory with the floor plan of *Maison Minimum* by LeCorbusier, which is encoded into hexadecimal numbers as FF803F71180EFE033F. This number and the two dimensioning sets X1 and X2 specify the plan completely.

When the same procedure is applied to three-dimensional forms, the envelope of a rectangular building is enclosed in a bounding box. The box is subdivided by a series of orthogonal planes that create an array of cuboids. Any cuboid that corresponds with a part of the built form is coded with an 1 while any cuboid that corresponds to an empty space is encoded with a 0. The 0s and 1s are listed in a single string. The cuboids are 'unpacked' by first separating out vertical slices in y, and then separating stacks of cuboids in x. In this way, the Seagram building is defined by the hexadecimal number 10083EFE0F00 and three dimensioning sets X1, X2, X3.

An interesting feature of the method is separation of the shape's topological and metric properties. The configuration of the built form is represented independent of its metric dimensions.

Philip Steadman adapts March's principle of binary encoding and develops a technique for representing a class of built forms (Steadman 1998) by applying a series of transformations to a generic form.

---

[1] "The word "isomorphism" applies when two complex structures can be mapped onto each other, in such a way that to each part of one structure there is a corresponding part in the other structure, where "corresponding" means that the two parts play similar roles in their respective structures." (Hofstadter 1980, p. 49)

Allowing for an arbitrary number of storeys, space is represented as being divided in three types of zones, distinguished by the nature of their lighting. The dimensionless configuration of the archetypal form can be represented as a matrix of cuboids, in which each court is represented as a single cuboid and strips of accommodation are represented by rows of cuboids.

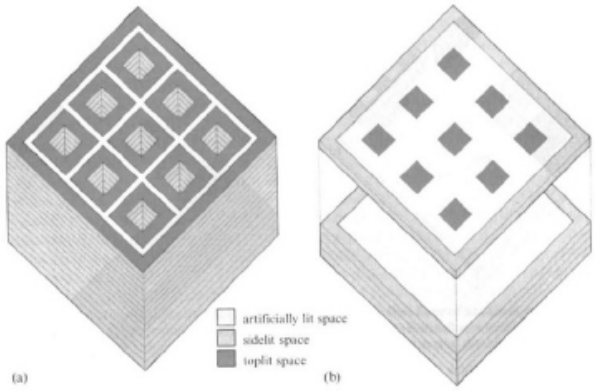


Figure 3. The archetypal built form (Steadman 2001)

Every configuration derived from the archetype might be described by a binary code that list all strips of accommodation in x and y and all floors in z. (Sreadman1998). In any particular form parts of the archetype which are selected for inclusion are designated by 1s, and parts which are suppressed by 0s. Then values can be set for all dimensions.

'Modular' approaches may be interesting, experimental methods of form description but they seem inadequate to address current requirements for complex architectural forms.

Representations in Evolutionary Design

Representation of form in strings is widely applied in Evolutionary Design techniques where the generative process operates with various parameters encoded into string-like structures.

Representational approaches used in the field of creative evolutionary design vary greatly. Some strings of values encode solutions and others rules on how to build solutions. (Bentley 1999).

Two innovative approaches to the layout problem of two-dimensional architectural floor plans are the learning approach developed by John Gero and the hierarchical growth approach developed by Mike Roseman (Roseman and Gero 1999). Both evolve designs by generating complex gene structures. The genes represent simple design actions which when executed produce parts of design solutions.

In the first case, coding constitutes a simple grammar for constructing orthographic shapes using turtle graphics. Four different basic genes either draw a line in the current direction, move the pen ahead, or change the current direction.

00: line forward, 11: step forward, 01: right turn, 10: left turn

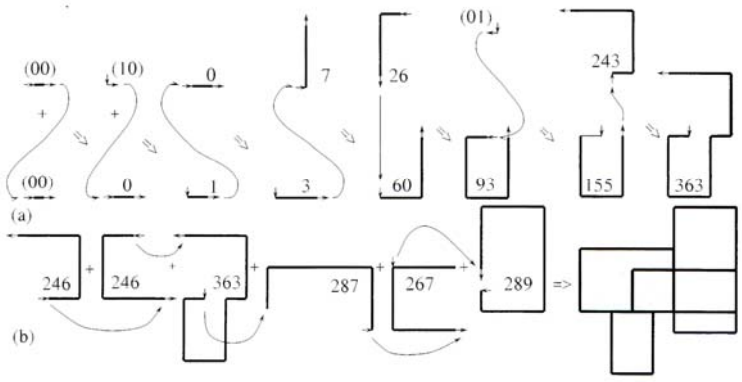


Figure 4. Evolved representation (Roseman and Gero 1999)

The second employs a design grammar of rules to define how polygons should be constructed based on a method for representing polygonal shapes as closed loops of edge vectors.

The phenotype of a polygon is the sequence of edge vectors which provides the description of that shape's structure. A suffix is used

to identify individual edges of the same vector type, e.g. (W1,N1,E1,S1). The genotype is the sequence of the two polygons used and the two edges joined.

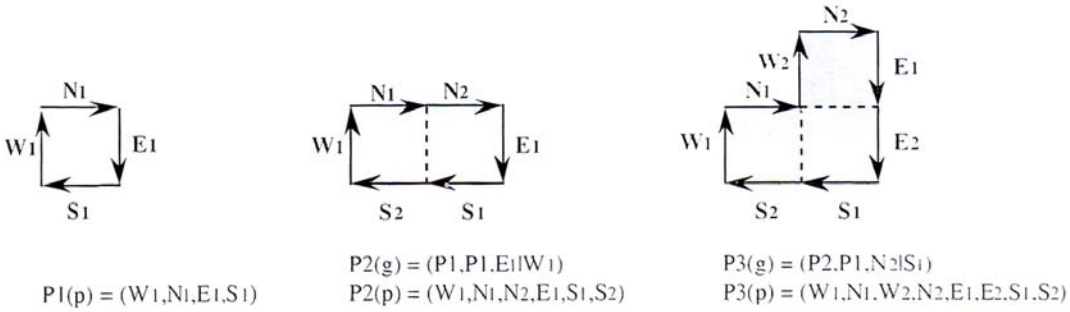


Figure 5. Generation of a trimino (Roseman and Gero, 1999)

Representations of 3dimensional form are usually based on modular assemblies of simple forms..

John Frazer developed in '69 "a densely coded description of a minimal configuration of the units to be developed and manipulated into a complex structural form without inputting any further data" (Frazer, 1995). The Reptile system consists of two structural units that can be given 18 different orientations relative to each other. The genetic code script is defined as the description of the unit in space in the form of (A,B,C,D,T',T'').

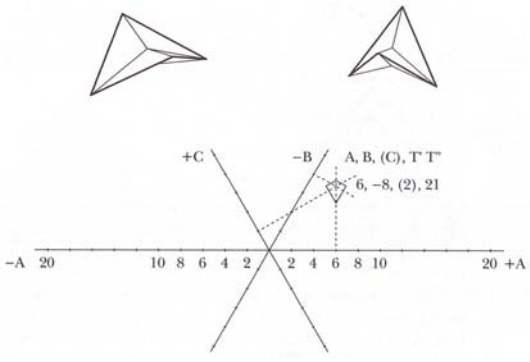


Figure 6. Location of units defined by (A,B,C, D,T',T'') (Frazer 2002)

In the Universal state space modeller he uses the isospatial grid and a system of close-packing spheres. Each mote has 12 neighbours of identical geometric conformation and equal center-to-center distance (Frazer 1995).

Paul Coates, Terence Broughton and Helen Jackson use Lindenmayer systems and genetic programming to produce recursively defined three-dimensional objects that satisfy certain goals. Form is represented through the insertion of spheres at the vertices of the isospatial2 grid. The symbol strings are made up by simple production rules which can be interpreted as a series of drawing instructions to produce an abstract representation of the organism. The rules include a) instructions to insert the sphere 'F', b) a positional variable indicating where to put the sphere 'POS1…POSn', c) an open bracket '(' that indicates a branching point and d) a closed bracket ')' which is the instruction to return to position on a lower 'limb' where it last branched.

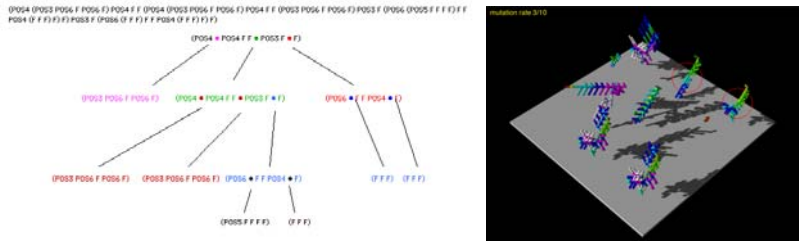


Figure 7. A genotype generated by the L-system and an evolved population (Coates, Brougthon and Jackson 1999)

Alternatively, Jackson discusses an embryology that results in orthogonal spatial configurations rather than isospatial forms makes use of an edge rewriting L-system is used, with an alphabet of 6 symbols (Jackson 2002).

[2] The isospatial grid is defined as a point and its 12 neighbours are defined by a dodecahedron. This repeats across 3d space jut as the orthogonal grid does, but without the 3 different point to point distances of the orthogonal grid. Bays and Frazer both have used this grid in cellular automata in place of the cubic grid. With 6 axes and 4 planes of symmetry it is a superset of the Cartesian grid.

Fl	Move forward a step and insert a rectangle of height: width ratio 1:3
Fr	Move forward a step and insert a rectangle of height: width ratio 3:4
PLUS	Rotate heading counterclockwise by 90° relative to current heading
MINUS	Rotate heading clockwise by 90° relative to current heading
( )	Brackets indicate branching points, allowing tree structures to be described

The rectangular forms are interpreted as plan representations of spaces

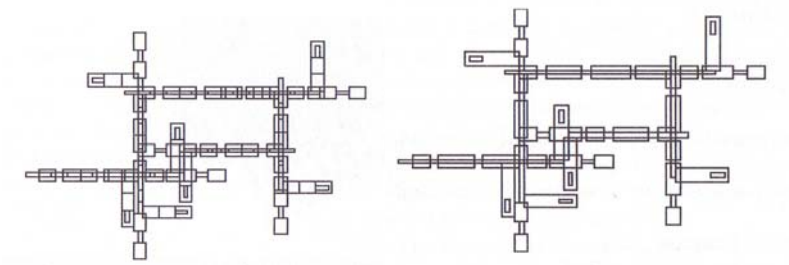


Figure 8. Genotype and interpreted phenotype (Jackson 2002)

Trying to find a generic representation for a system capable of designing a wide variety of different designs, Bentley and Wakefield developed 'clipped stretched cuboids' (Bentley1996).

The representation uses a number of combined primitive shapes, which consist of a cuboid with variable width, height and depth, and variable three-dimensional position. Each cuboid can be intersected by a plane of variable orientation, to allow the approximation of curved surfaces. The cuboid's geometry is defined by a nine parameters. Designs are defined by a number of non-overlapping cuboids.

The genotype consists of a single chromosome arranged in a hierarchy consisting of multiple blocks of nine genes. This arrangement corresponds to the spatial partitioning representation

used to define the phenotypes, with each block of genes being a coded primitive shape and each gene being a coded parameter.

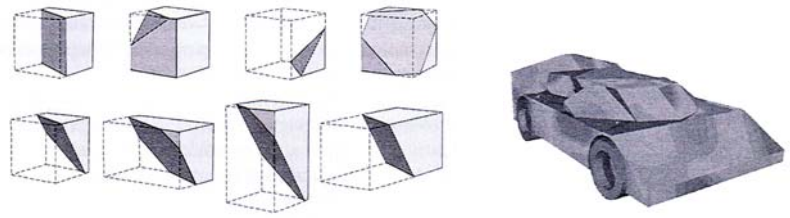


Figure 9.Clipped stretched cuboids in generic evolutionary design (Bentley 1999)

An example of form representation in Evolutionary Art is Todd and Latham's *Form Grow*, where recursive grammar rules using constructive solid geometry take a series of real numbers as parameters.

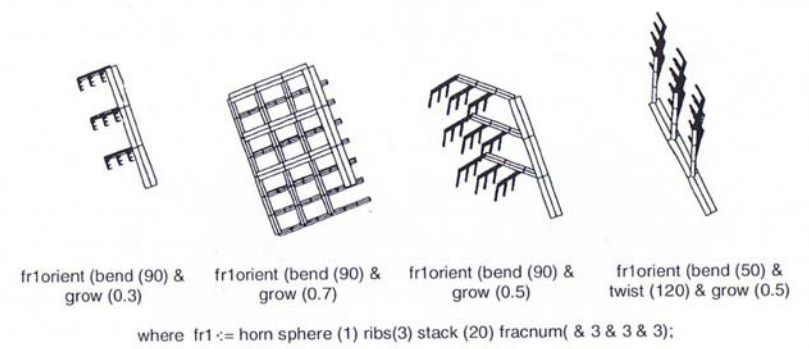


Figure 10. *Form Grow* rules and forms (Todd and Latham 1999)

The basic construction takes a number (how many times the input form will appear in the horn) of input forms, an input form (defines the objects out of which the construction is to be built) and a list (defines how and where each element will be arranged using operations of transform rules (such as twist/bend/stack/grow). The system takes a structure expression and generates a corresponding starting gene vector.

In most of the above mentioned representations, the values in the strings can be defined independently of each other; there is no underlying connection between them. They are not related, they just specify irrelevant quantities. Having instead values that are related in a certain way comes into the area of Parametric Design.

Parametric design

In parametric architectures it is the parameters of a particular design that are declared, not its shape. Parametric Design turns 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, i.e. parameters, which can be independent or dependent. When the parameters are assigned specific values, particular instances are created from an infinite range of possibilities. (Kolarevic 2003)

Parametric design implies the use of parameters to define a form but of greater importance are the underlying relations between elements of the form. The set of equations establishes relationships between objects which are maintained while the elements can be independently modified. When interdependencies between objects are established, the objects' behaviour under transformations is efficiently defined. Relationships can also be revisited and revised during the design process.

Mark Burry (Burry 2003) observes that, at the moment, parametric design refers to Cartesian geometry and "the ability to modify the geometry by means other than erasure and 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. "

A parametric description of form provides is flexible enough to represent complex curves and surfaces. This can be perfectly demonstrated by Burry's extended application of parametric

techniques on the analysis of the ruled surfaces used by Antonio Gaudi. He applies parametric design software at the Sagrada Familia with the intention "to remodel and resolve the use of these surfaces into a measurable interpretation that can be used to advance the building work" (Burry 2001).

The focus of attention centres on a special type of surfaces, ruled surfaces, and their characteristics. Being easy to define and construct at a local level, ruled surfaces are able to accomplish high levels of form complexity, especially by their intersections when assembled.

Ruled surfaces have been extensively applied to architecture, with their potential adjusted to the new technological means. The Paramorph, designed by DeCOi in 1999, may serve as an example.

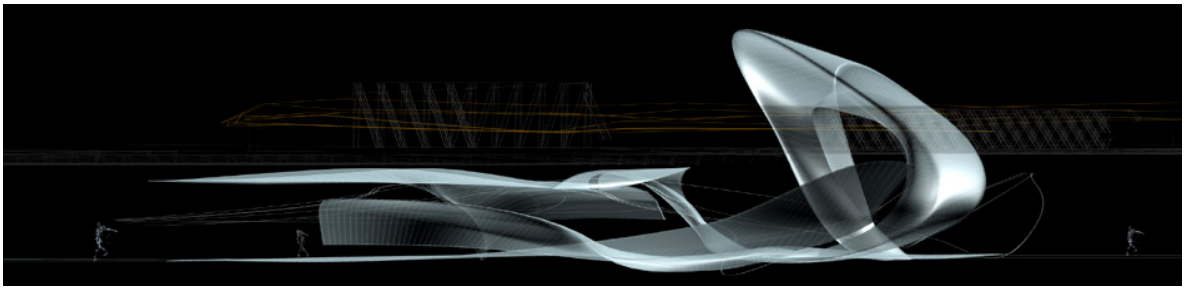


Figure 11. Paramorph (DeCOi 2000)

An extensive presentation of ruled surfaces and their qualities follows in the next chapter.

Ruled surfaces

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*

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

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

Curve *b* is called the *directrix* or ***base curve*** of the surface. In other words, we get the *director 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.*

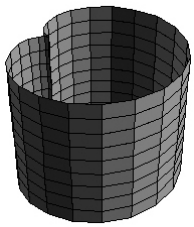
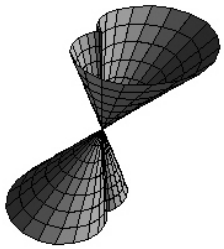
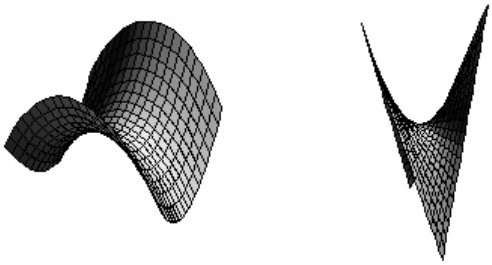
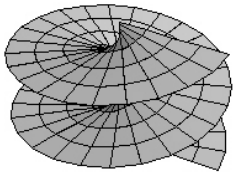
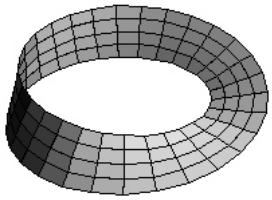
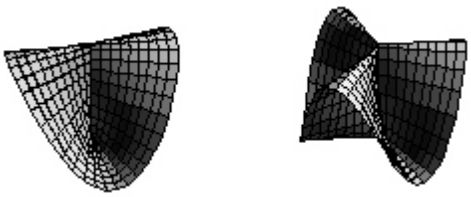
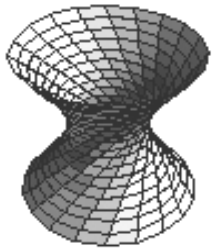
A surface is called a *ruled surface*, if it has a C^2-parametrization of the following kind:

$$f(u,v) = b(u)+v\ .\ X(u)$$

where *b* is a (differentiable, but not necessarily regular) curve and *X* is a vector field along *b* which vanishes nowhere. (Kühnel 2002)

It is clear from the expression that the v-lines (with constant u) are Euclidean lines in space. They are called *generators* or ***rulings*** of the surface and can be intuitively understood as the various positions of the generating line *L*, the movement of which depends on one parameter. Frequently, it is necessary to restrict v to some interval, so the rulings may not be entire straight lines.

The simplest ruled surface is the PLANE. Other examples include:

<table>
<tr>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>HYPERBOLIC PARABOLOID or SADDLE
(a doubly ruled surface)
 
its vertical cross sections are parabolas, while the horizontal cross sections are hyperbolas.</td>
<td>HELICOID
 
MÖEBIUS STRIP
  </td>
</tr>
<tr>
<td>PLÜCKER'S CONOID (or CYLINDROID)
  </td>
<td>HYPERBOLOID OF 1 SHEET
a doubly ruled surface
  </td>
</tr>
<tr>
<td colspan="2">Images from MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com</td>
</tr>
</table>

The position vectors of the following surfaces can be expressed parametrically.

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	$\begin{bmatrix} a(u+v) \\ \pm b v \\ u^2 + 2 u v \end{bmatrix} = \begin{bmatrix} a u \\ 0 \\ u^2 \end{bmatrix} + v \begin{bmatrix} a \\ \pm b \\ 2 u \end{bmatrix}$
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	$\begin{bmatrix} r \cos\theta \\ r \sin\theta \\ 2 \cos\theta \sin\theta \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \cos\theta \sin\theta \end{bmatrix} + r \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}$

Generating a ruled surface

Ruled surfaces may be generated or defined in several ways. Arnold Emch (Emch 1919) refers to five distinct ways of generating the surfaces (appendix I).

In CAD packages: Given two curves $\boldsymbol{C}_1(u)$ and $\boldsymbol{C}_2(v)$, the ruled surface is the surface generated by connecting line segments between corresponding points, one on each given curve. More precisely, if t is a value in the domain [0,1] of both curves, a segment between $\boldsymbol{C}_1(t)$ and $\boldsymbol{C}_2(t)$ is constructed, the ***ruling***. As t moves from 0 to 1, the ruling at t sweeps out a surface and this is the ruled surface defined by curves $\boldsymbol{C}_1(u)$ and $\boldsymbol{C}_2(v)$.

(from http://www.cs.mtu.edu/~shene/COURSES/cs3621/LAB/ surface/ ruled.html)

According to Stephen Ervin and Hope Hasbrouck (Ervin and Hasbrouck 2001), the type of input entities for the ruled surface varies according to the software package, but can be generated from any combination of the following: point, line, polyline (open or closed). Input entities should be at the appropriate z elevation and must have similar topological directions. There is only one exception: a ruled surface cannot be generated between an open and a closed polyline, or between two points.

They present the procedure as follows:

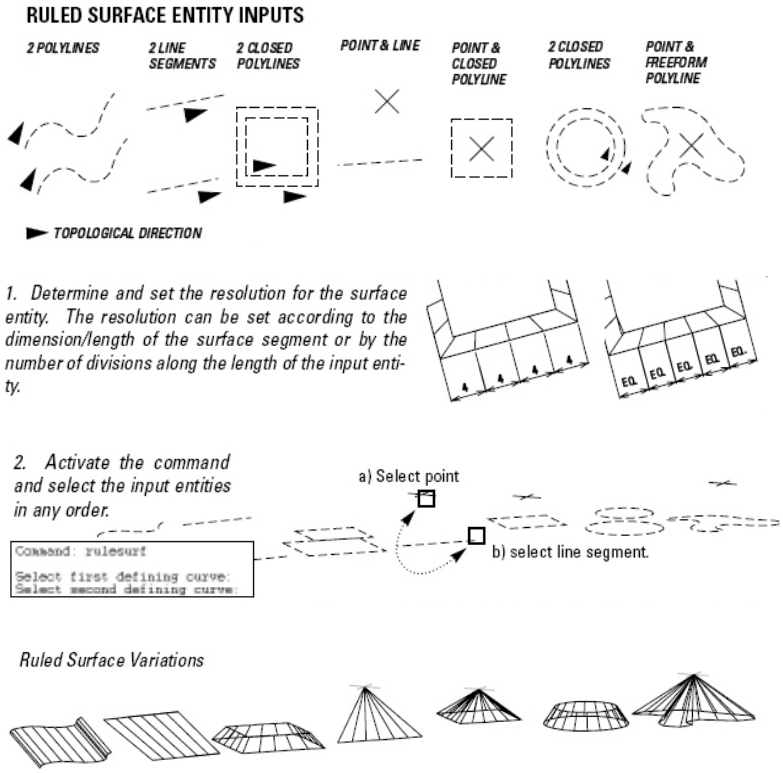


Figure 12. Construction of a ruled surface (Ervin and Hasbrouck 2001, p58)

Architectural Structures

The first engineer that designed hyperboloid structures was Vladimir Shukhov, at the end of 19th century. Shukhov derived a family of equations that allowed the construction of new structural systems: hyperboloids and hyperbolic paraboloids. The doubly-curved surfaces of the roof structures and water towers he designed were formed of a lattice of straight angle-iron and flat iron bars.[3]

Figure 13. The world's first hyperboloid water tower, All-Russian Exposition, Nizhny Novgorod, Russia, 1896 (Images from Wikipedia, http://en.wikipedia.org/)
Figure 14. Shukhov Radio Tower, Moscow, 1919-22

During 1880-1895, Gaudi and Shukhov were experimenting with hyperboloid structures simultaneously but independently. Gaudi's application of ruled surfaces was more extended. He used three surfaces that could be constructed at his time: hyperbolic paraboloid, helicoid and hyperboloid, all generated through straight lines. The entire design of the nave in the Sagrada Familia is an assembly of these three geometries. (Burry 1993)

[3] By 1918 Shukhov designed a 350m radio transmission tower for Moscow, which would have surpassed the Eiffel tower in height by 50m, while using less than a quarter of the amount of material. The tower was never built due to lack of the 2200 tons of steel required. A 150m tower was constructed instead, in 1922.

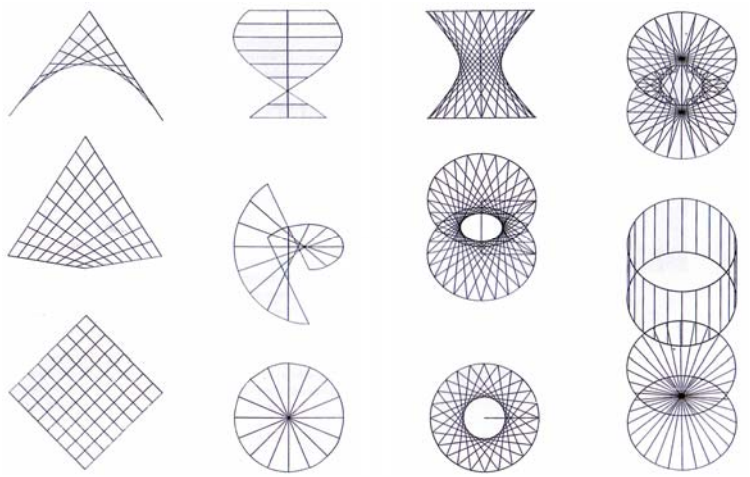


Figure 15. Ruled surfaces used in Sagrada Familia (Burry 1993)

Felix Candela has also extensively used these forms, although in his case their use tend to be more singular.

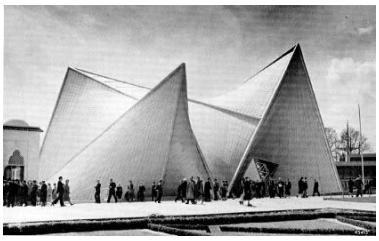


Figure 16. Phillips Pavilion for Brussels World Fair, designed in 1958 by Le Corbusier, was based on an assembly of hyperbolic paraboloid shells.

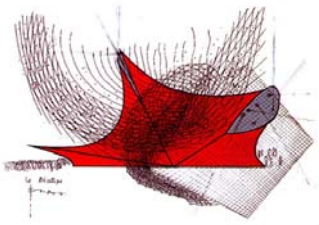


Figure 17. According to Robin Evans (Evans) ruled surfaces lie beneath the design of Ronchamp Chapel.

(Images from Wikipedia, http://en.wikipedia.org/)

Special features

Ruled surfaces are attractive from a mathematical, aesthetical and structural point of view.

Their application in Art demonstrates their aesthetic appeal. They were considered as symbols of scientific imagination.
Since 1941 Naum Gabo and then his brother Antoine Pevsner had been making "linear constructions" by stretching threads or wires over a rigid formwork to imply continuous warping surfaces.

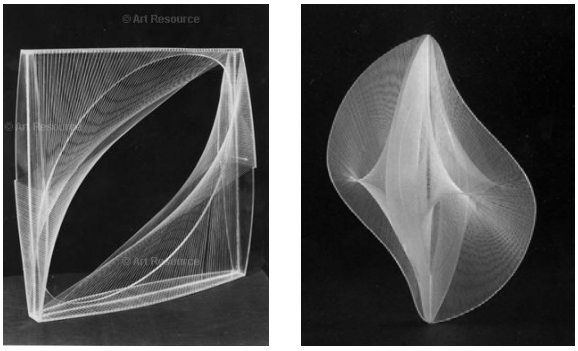


Figure 18. Linear Constructions (Images from www.tatemodern.org)

Iannis Xenakis –who was also involved in the design of the Philips Pavillion- used ruled surfaces as an alternate system of musical notation in a chamber orchestral piece entitled *Metastasis* in 1954. (Evans, 1995).

Their most important feature is their simplicity of construction relative to their striking shape.
Burry (Burry 1993) emphasizes the ability of those surfaces to facilitate construction: "This apparently plastic, organic architecture is yet an architecture composed of surfaces which can be more readily described stereotomically than can some primary geometries. Any of these surfaces can be simply enough *described* as the points of origin and termination of an appropriate number of straight lines, which, when interpreted by the masons chisel, blend

sufficiently to form the desired surface with great precision[4].” Furthermore, the components of a larger whole can be carved independently off-site and then be assembled to form the final composition.

Ruled surfaces are widely applied in two-dimensional digital fabrication -CNC cutting-, where production strategies involve extraction of two-dimensional, planar components from geometrically complex surfaces or solids comprising the building’s form. In more detail “One method of “rationalising” double-curved surfaces is to convert them into “rule-developable” surfaces. They are fairly easy to construct using conventional construction techniques….they are used extensively in contemporary architectural practice because they can be “developed”, i.e unfolded into flat shapes in modelling software, and digitally fabricated out of flat sheets. (Kolarevic 2003)

Additionally, of particular interest is the close relationship of seemingly dissimilar geometrical forms. The movements which the various parts admit, cause one surface gradually to pass into another as in a "dissolving view" so that, for example, a plane seen to be only a particular case, or limit, of a hyperbolic paraboloid, a cylinder and a cone to be only extreme cases of a hyperboloid of one sheet.

[4] The mason is provided with templates for the joints of particular stones, (or of edges resulting from the intersection between two distinct surfaces) which are marked to indicate the origin and destination of the straight lines that form the surface. The chisel is used to connect the corresponding points.

3. Aim and objectives

Based on the general framework presented, a parametric system of representation is developed that attempts to satisfy the need for an innovative way of representing architectural form.

This thesis proposes that an efficient system should be structured based on the relationships between elements that constitute the form. Representation can be outlined as choosing the right parameters and establishing connections between them. The architectural structure to be represented can be nothing more than a series of values which define its form.

The system should respond to up to date architectural and constructional advances by being able to represent both planar and curvilinear forms. Ruled surfaces seem suitable to be used as the basic elements of a parametric system of representation, as one of their inherent qualities is two-dimensional simplicity that can produce three-dimensional complexity. Moreover, when building blocks are surfaces are composed of straight lines static calculations as well as the actual construction are simplified. The units offer indications of the structure and its behaviour.
The system should also be simple enough, developed for a limited number of parameters, user-friendly and efficient in terms of speed.

Encoding a structure in this way makes possible its integration in evolutionary design techniques, such as genetic algorithms. Design experiments so far have been quite successful although they have not reached a required level of three-dimensional form generation, partly because of the lack of an appropriate representation.

By developing such a system the thesis attempts to address a wider context of enquiry.

- It explores the power of parametrics. What are the advantages offered when design is based on interrelations of elements?
- As Kolarevic mentions “representation actively shapes the designer’s thinking process.” Can innovative ways of representing form contribute in enhancing the design process? In what ways?
- Can a system of representation extend the role of representation in the design process? Can it be actively integrated in the process of form generation?

To answer these questions a simple parametric system based on ruled surfaces was created, according to the criteria that were pointed out. A program was constructed in Processing language so that the system could be tested against real buildings. The system was compared to previous approaches of representation and the results were evaluated.

4. Methodology

Development of the parametric system

In the general framework of computer-aided design the process of defining the geometrical form of a building is generally one of composing elementary forms together (Steadman and Waddoups). Reversing this statement, an architectural representation could be based on the decomposition of the structure into elementary units, in this case a number of surfaces.
Given that a number of parameters can sufficiently define the form of each surface, each building can be represented as a series of values for the parameters of every surface, presented in a matrix.

The system's development started from the study of the structure's elementary units, the surfaces. Once the surfaces were generated, they could be assembled to form the architectural composition.
The ruled surfaces had to be modelled 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's investigation. The number of parameters was kept the lowest possible, as it is observed that a small number of variables create a wider range of solutions than a higher one (Krishnapillai).
The use of parametric equations was obviously favoured against algebraic expression of the surfaces. Mathematical equations were expressed as position vectors.

The most common way of constructing the surfaces in CAD packages is using lines that join corresponding points between two algebraic curves. However, the most common way of describing the surfaces in mathematical terms is with a straight line moving

along a curve with the direction given by another curve. The latter was selected as more appropriate.

The construction of a surface required two parameterised 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 generatrix). 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 parameterised curve *c* so that it points in the direction *z(t)* at each time t.

In theory every arbitrary free-form curve can produce a ruled surface. In the simplest case this is a surface produced by the curve's extrusion in the direction of a straight line. Initial considerations of using free-form curves were soon discarded, although they produce a wider variety of surfaces. Since the objective was the representation of built structures, the attention shifted to those surfaces that had already been repeatedly used in architecture. In general, *cylindrical* and *conical surfaces* have been of the broadest use. The *hyperboloid* and the *hyperbolic paraboloid* are the most typical examples of ruled surfaces, followed by the *helicoid* and the *moebius strip*.

Looking at the parametric functions of those surfaces, it was observed that most of them are expressed in terms of the sine and cosine functions and appear to have an underlying relationship. The relationship 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]

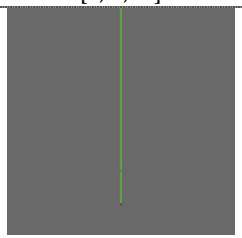
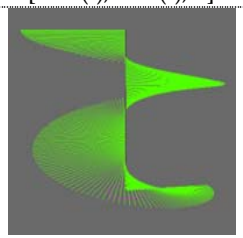
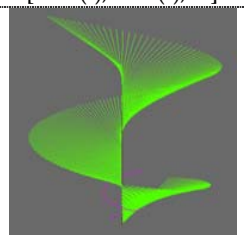
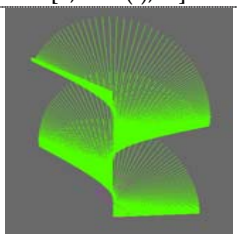
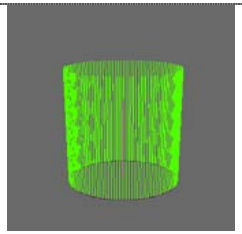
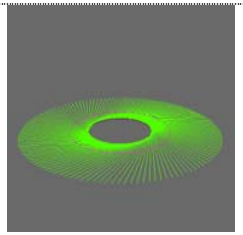
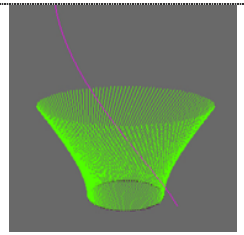
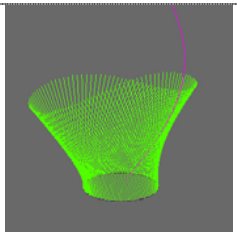
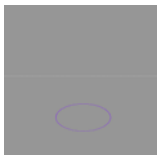
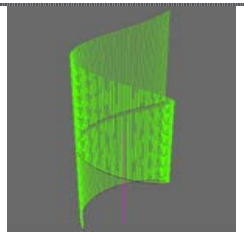
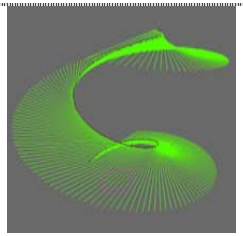
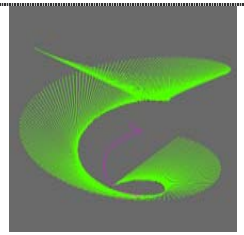
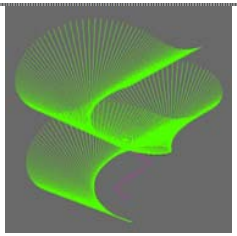
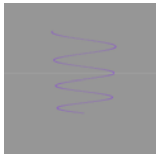
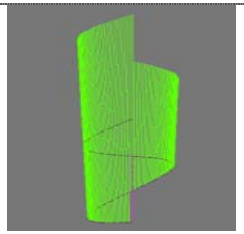
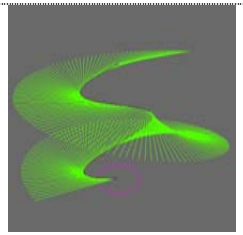
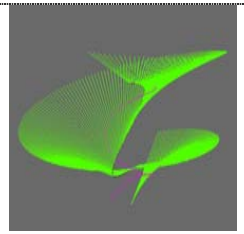
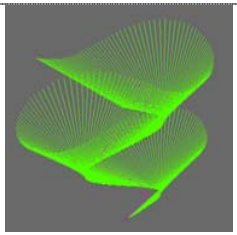
If the z vector component is suppressed the curve is degenerated to a **circle**.
If the x vector component is suppressed the curve is degenerated to the parametric **sine** function.
If the x and y vector components are suppressed the curve is degenerated to a **line**.

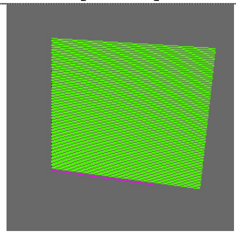
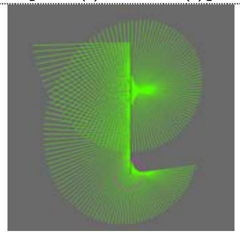
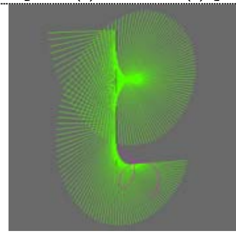
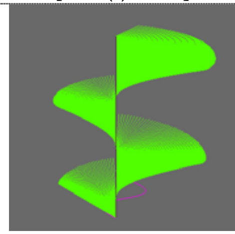
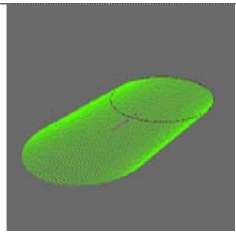
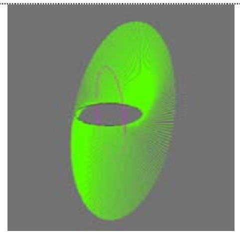
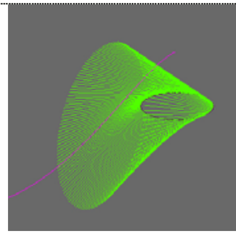
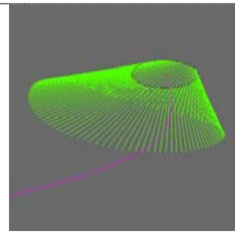
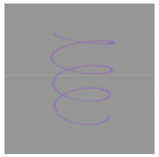
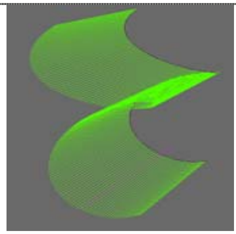
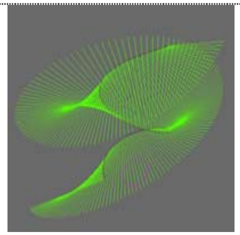
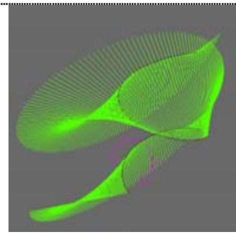
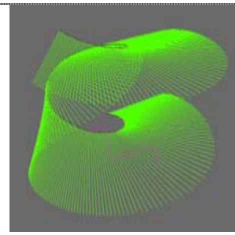
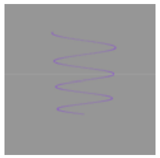
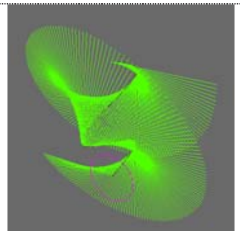
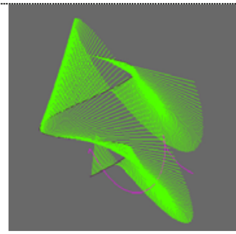
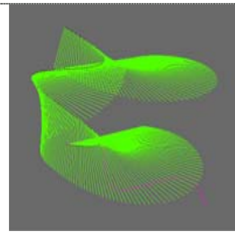
Curves represented as Vectors of trigonometric functions	
Circle	[a*cos(t), b*sin(t), 0]
Helix	[a*cos(t), b*sin(t), c*t]
Sine	[0, b*sin(t), c*t]
Line	[0, 0, c*t]

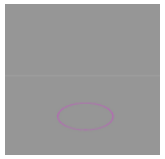
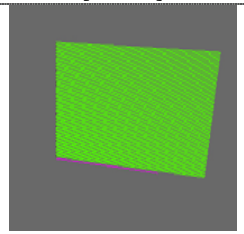
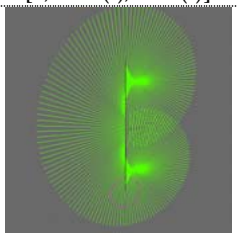
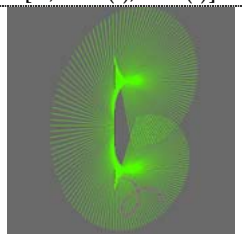
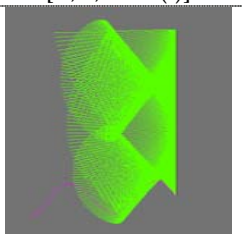
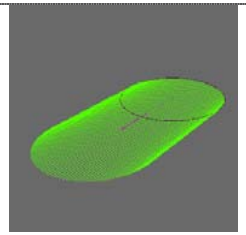
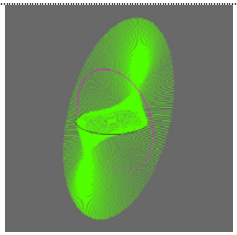
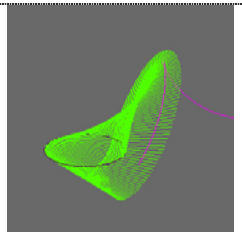
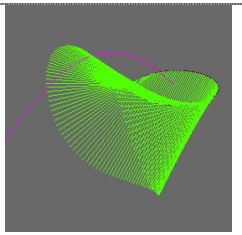
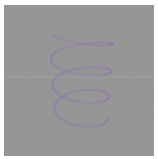
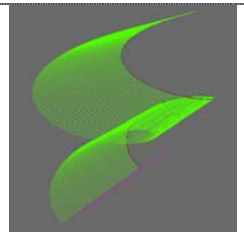
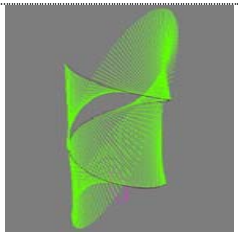
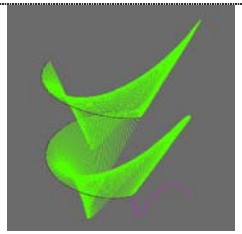
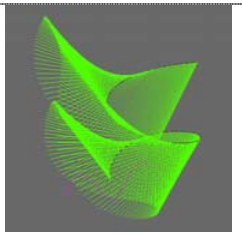
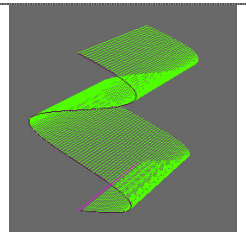
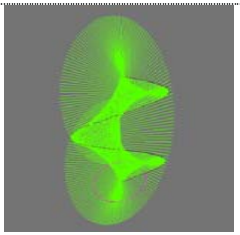
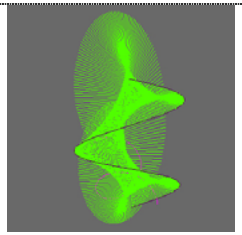
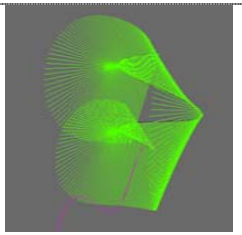
The combination of these four curves as base and director respectively can generate sixteen ruled surfaces. The rotation of one curve around the axes generates thirty two extra surfaces (sixteen for each rotation).

Among these forty eight surfaces the plane, the cone, the cylinder, the hyperboloid and the helicoid 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 three tables. In all tables, the base curve is constantly oriented along the z-axis while the director curve is rotated each time.

		DIRECTOR CURVE [*a*cos(t), *b*sin(t), *c*t]			
		Line	Circle	Helix	Sine
		[0, 0, *c*t]	[*a*cos(t), *b*sin(t), 0]	[*a*cos(t), *b*sin(t), *c*t]	[0, *b*sin(t), *c*t]
BASE CURVE					
					
					
					

		DIRECTOR CURVE [*b*sin(t), *c*t, *a*cos(t)]			
					
		Line	Circle	Helix	Sine
		[0, *c*t, 0]	[*b*sin(t), 0, *a*cos(t)]	[*b*sin(t), *c*t, *a*cos(t),]	[*b*sin(t), *c*t, 0]
BASE CURVE					
					
					
					

	DIRECTOR CURVE [*c*t, *a*cos(t), *b*sin(t)]			
				
	Line	Circle	Helix	Sine
	[*c*t, 0, 0]	[0, *a*cos(t), *b*sin(t)]	[*c*t, *a*cos(t), *b*sin(t)]	[*c*t, 0, *b**sin(t)]
BASE CURVE					
					
					
					

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

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- was 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 12xn matrix can represent a building.

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

where:

α_b, b_b, c_b } parameters of [a*cos(t), b*sin(t), c*t] for base curve

α_d, b_d, c_d } parameters of [a*cos(t), b*sin(t), c*t] for director curve

t_x : translation of surface around X axis

t_y : translation of surface around Y axis

t_z : translation of surface around Z axis

r_x : rotation of surface around Z axis

r_y : rotation of surface around Z axis

r_z : rotation of surface around Z axis

n: total number of surfaces that compose the represented building

Development of the program

Testing the system involved the construction of a program in Processing language that draws the composition of surfaces. The program receives a series of parameters for each surface as input by the user and draws an assembly of surfaces that represent the building.

Draw a surface

First a short program that draws a surface was developed.
For each ruled surface the base curve is defined by the vector

$$[a_b*\cos(t) , b_b*\sin(t), c_b*t]$$

The director curve is defined by the vector

$$[a_d*\cos(t) , b_d*\sin(t), c_d*t]$$

The plane construction requires two lines along different axes. If the base curve is defined as mentioned above, the director curve should be defined either as

$[b_d*\sin(t), c_d*t , a_d*\cos(t)]$ or $[c_d*t , a_d*\cos(t), b_d*\sin(t)]$

The method that was selected was adding an extra value in the x component of the director vector so that it becomes

$$[a_d*\cos(t)+ d_d*t , b_d*\sin(t), c_d*t]$$

The program receives the parameters a_b, b_b ,c_b and a_d, b_d, c_d, d_d as input and executes the following algorithm.

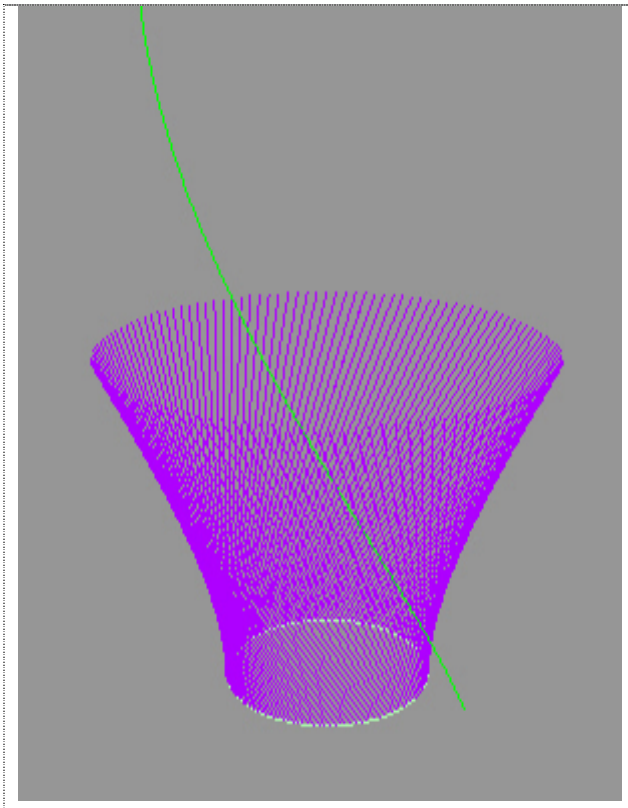
Algorithm 1

At each step

1. *increment parameter t by a constant number*
2. *calculate base and director curve vectors for value t and draw a point with coordinates defined by each vector*

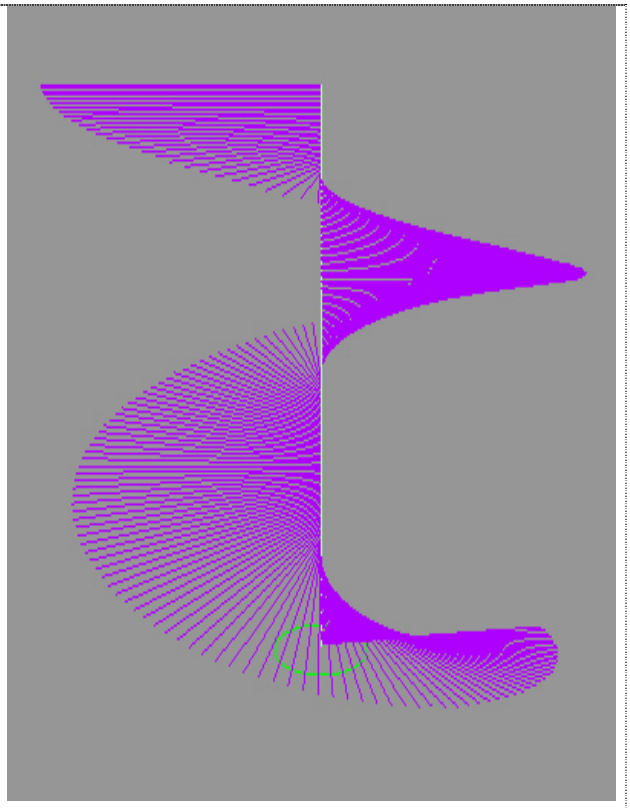
3. *join consecutive points to draw two curves*
4. *calculate slope of director curve by subtracting successive values for each vector component.*
5. *normalise slope vector*
6. *draw a line of fixed length starting from the point on the base curve and has its direction defined by the slope of the director curve, i.e a ruling.*

This iterative process is repeated for every point of the base curve. A new line is generated for each value of parameter *t* (appendix IIa). The assembly of those lines compose the surface.



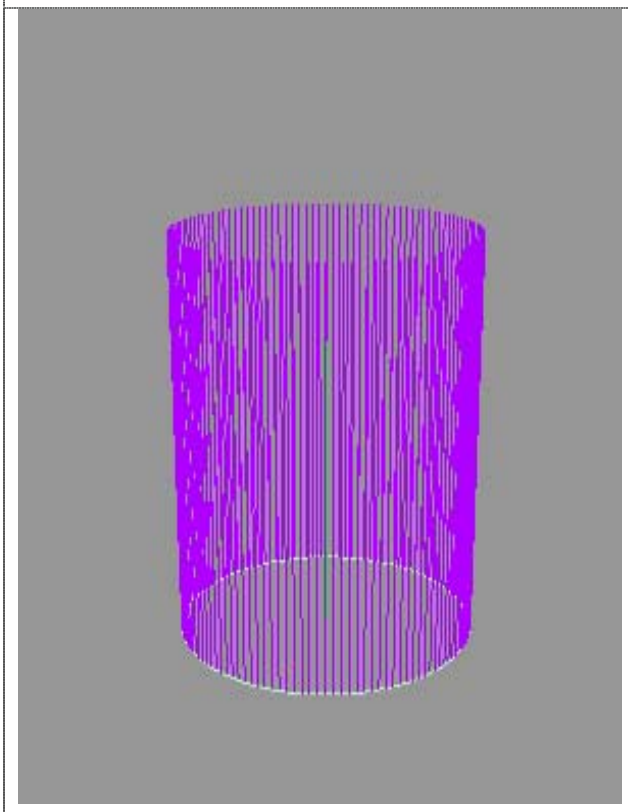
$\alpha_b = 25$, $b_b = 25$, $c_b = 0$ -circle - *HYPERBOLOID*

$\alpha_d = 30$, $b_d = 30$, $c_d = 40$ -helix-



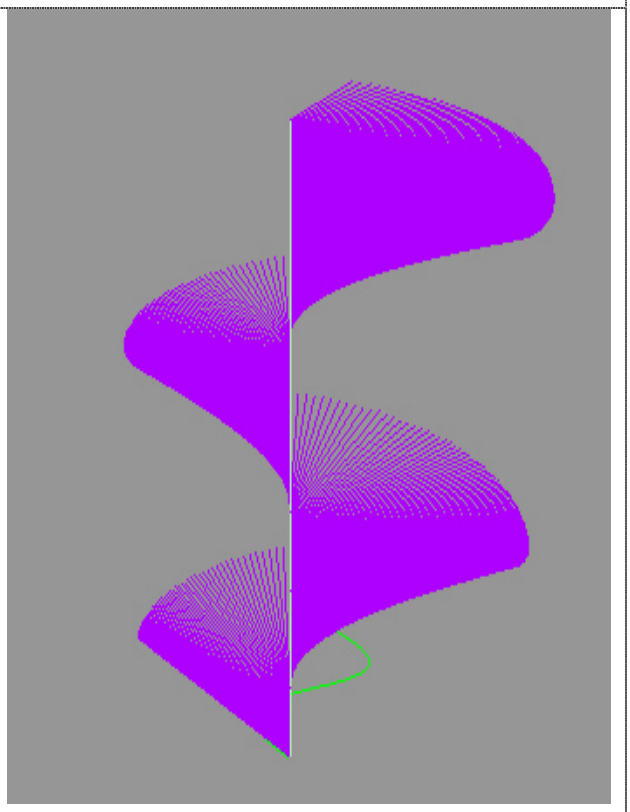
$\alpha_b = 0$, $b_b = 0$, $c_b = 25$ -line- *HELICOID*

$\alpha_d = 30$, $b_d = 30$, $c_d = 0$ -circle-



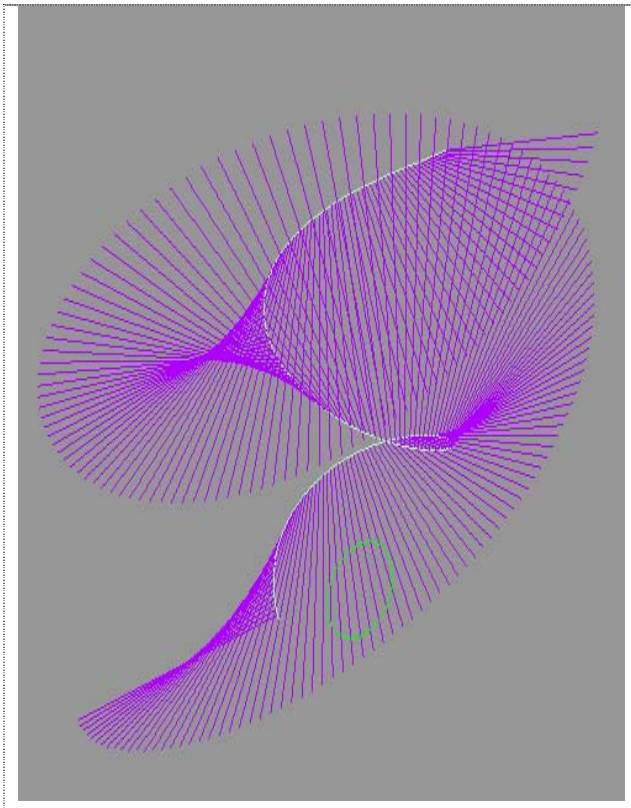
$\alpha_b = 30$, $b_b = 30$, $c_b = 0$ -circle- *CYLINDER*

$\alpha_d = 0$, $b_d = 0$, $c_d = 10$ -line-



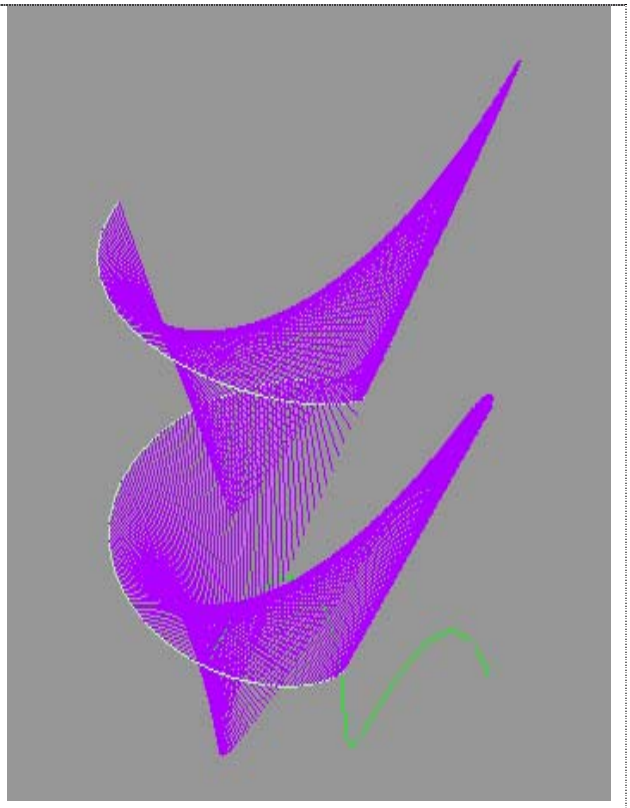
$\alpha_b = 0$, $b_b = 0$, $c_b = 25$ -line-

$\alpha_d = 0$, $b_d = 30$, $c_d = 10$ -sine-



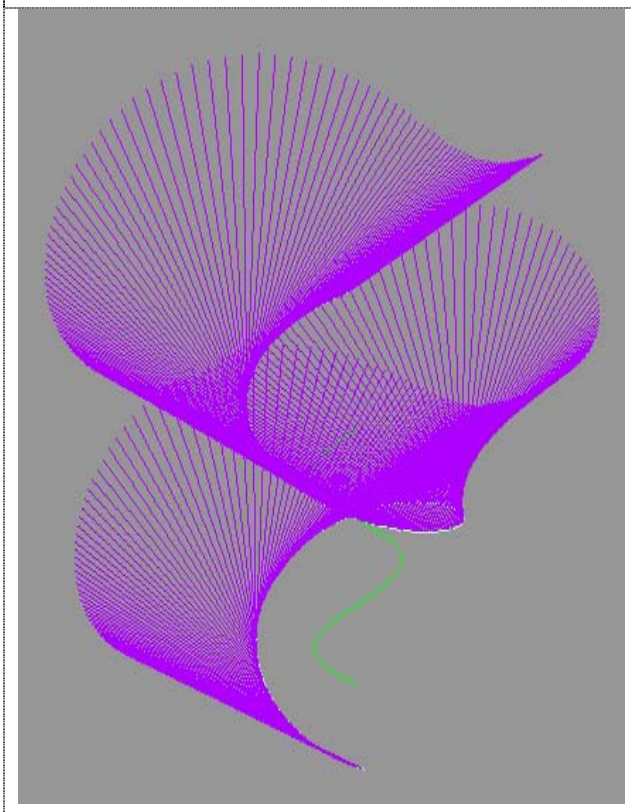
$\alpha_b = 50$, $b_b = 50$, $c_b = 25$ -helix-

$\alpha_d = 30$, $b_d = 30$, $c_d = 0$ -circle-



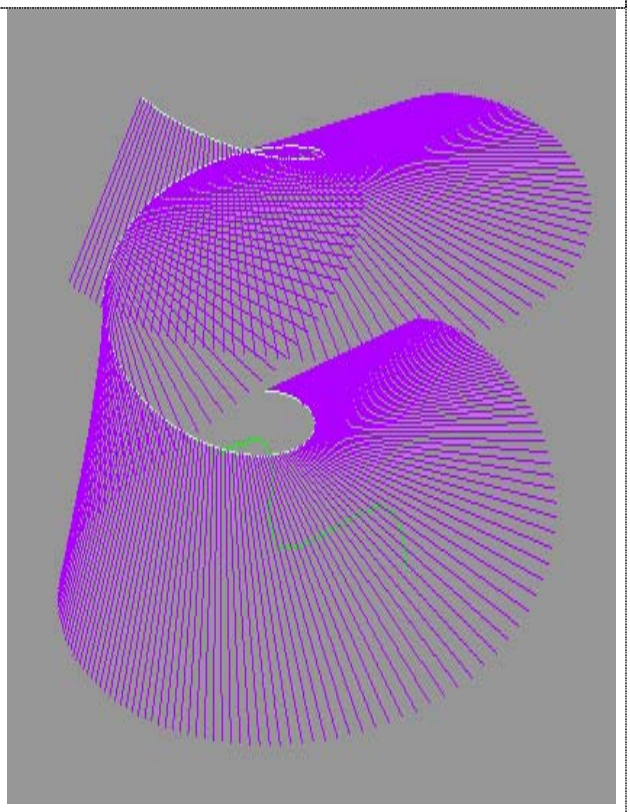
$\alpha_b = 50$, $b_b = 50$, $c_b = 25$ -helix-

$\alpha_d = 30$, $b_d = 30$, $c_d = 15$ -helix-



$\alpha_b = 50$, $b_b = 50$, $c_b = 25$ -helix-

$\alpha_d = 0$, $b_d = 30$, $c_d = 10$ -sine-



$\alpha_b = 50$, $b_b = 50$, $c_b = 25$ -helix-

$\alpha_d = 0$, $b_d = 30$, $c_d = 10$ -sine-

Draw the assembly of surfaces

After one surface was successfully generated the assembly of many surfaces into a unified whole was the next step considered. The code was modified to include the surface as a "class", so that multiple instances can be created simultaneously. As long as the values of the parameters for each surface vary, the instances created display a wide variety of forms.

The vectors that define each surface can describe its inherent relations but not its location in the general framework. The result is that each instance is drawn at the same location. Transformations were required in order to place every new surface in the desired location.

At this point two alternatives were considered, either locating each new instance depending on the global coordinates system, or using a local coordinate system that draws a surface in relation to the previous surface[5].
In any case, six more variables are required to define the transformation matrices - translateX, translateY, translateZ rotateX, rotateY, rotateZ -.

The way to accomplish the specific task was by applying each one of the transformations to the vectors of the two curves and recalculating them. The curves are translated and rotated, not the surface.

Because computer graphics apply the transformations one after the other the algorithm used was the following:

---

[5] The local coordinate system was initially selected, but it proved to be inadequate in the next stage, when intersections had to be calculated.

Algorithm 2

At each step

1. *Calculate base and director position vectors for value t*
2. *apply translation in 3 axes, calculate new vectors*
3. *apply rotation in axis X, calculate new vectors*
4. *apply rotation in axis Y, calculate new vectors*
5. *apply rotation in axis Z, calculate new vectors*
6. *use the vectors from previous step as base and director vectors*

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 (appendix IIb).

Several values are calculated for the properties of each line, thus allowing precise control of the relation between surfaces. For example, if the end point of ruling 5 of surface 1 is input as the translation coordinate Y for surface 2, the base curve of surface 2 will be located at the end point of ruling 5 of surface 1 (appendix IIc).

Intersections of surfaces

Once the surfaces are assembled together, the intersections of the surfaces had to be computed based on the properties of the rulings. The method that was followed was defining the planes which are formed between successive lines and calculating the intersection point between each line of one surface with each plane of the other surface (appendix IId).

After the point of intersection between a ruling and a surface had been calculated, the ruling was 'trimmed' by substituting its end point with the intersection point.

Algorithm3

For each line

1. *define plane formed between two ends of line and beginning of next line by calculating its normal.*

For each plane on surface1 and each line of surface2

2. *calculate value u to obtain coordinates of intersection point*
3. *substitute end of line[i] with intersection point[i] in order to trim the line*

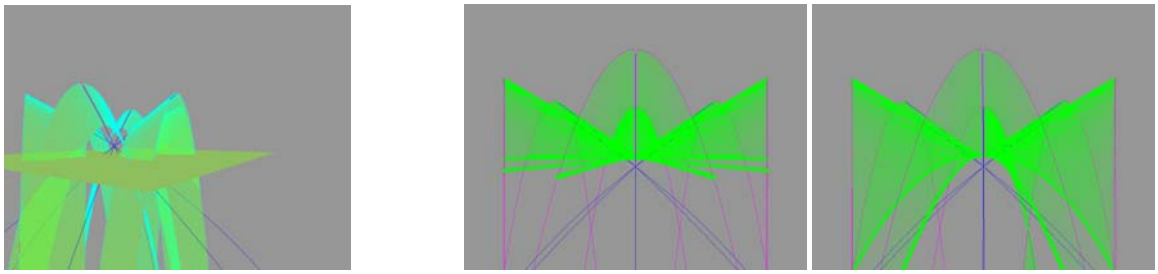


Figure 19. Intersections between a) surface and horizontal plane and b) two surfaces

Hyperbolic Paraboloid

The hyperbolic paraboloid, probably the most interesting ruled surface applied in architecture, didn't appear in the tables. Its construction requires the use of another type of curve, the parabola, the function of which is expressed by the position vector [t, 0, t*t].

Although the number of parameters the system takes as input is required to be kept at the lowest number possible, it was tempting to extend the variables in order to generate the additional form. The program was modified by adjusting the general functions for the base and director curves.

Two alternative expressions are generally used to define the surface.

- Base curve: line / director curve: parabola
- Base curve: parabola / director curve: parabola

The first expression requires less additional parameters.

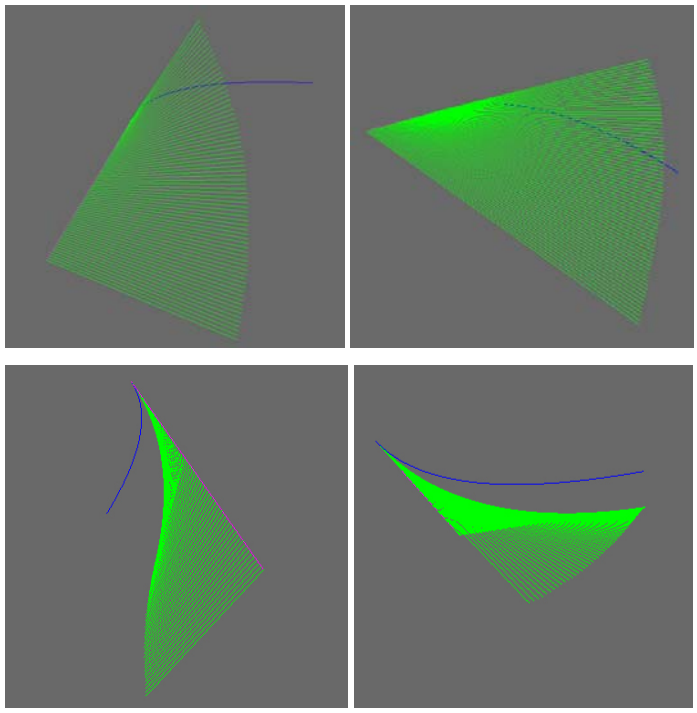


Figure 20. Hyperbolic Paraboloid in Processing.

general functions:

base curve: [a_d*cos(t)+ d_d*t , b_d*sin(t), c_d*t]

director curve: [a_d*cos(t)+ d_d*t , b_d*sin(t), c_d*t+e_d*t*t]

e_d : extra parameter added to the vector's components e_d

Representation of real buildings

The program was then implemented on a selection of existing buildings. It was preferred to test the system on a wider variety of buildings and obtain a compact representation than representing a complex building in more detail. The result of having simple representations with decreased precision was counterbalanced by understanding the range of possibilities for different building types.

The intention was to explore the suitability of ruled surfaces as units for both the exterior and the interior of a building. The application of the system was initially focused on the envelope of the building as it was assumed that the same principles will not fail to describe the internal division of spaces.

Five architectural structures of various types and forms were selected and represented.

Bodegas Ysios has a simple but appropriate form. The load - bearing walls trace a sinusoidal shape and the roof is a ruled surface wave, which combines concave and convex surfaces as it evolves along the longitudinal axis.

The traffic control tower of El Prat combines cylindrical and conical surfaces with a hyperboloid.

Palace of Assembly is a more complex synthesis of different surfaces: hyperboloid, planes, cylindrical surface in addition to a pyramid.

Los Manantiales is a illustrative sample of an assembly of hyperbolic paraboloids rotated around a central point. It was selected to test the extended version of the program.

The *Seagram Building* is composed of planar surfaces. It was selected for a comparison to the representation proposed by March.

The values of the parameters are presented in 13xn matrices, with the exception of the hyperbolic paraboloids assembly.

The matrices vary in size, they are 13x*n* 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.

After attempting to represent the buildings, the effect of changing values for the parameters was investigated.

It should be noted that the quality of the images can easily be misleading, and in this case doesn't do justice to the system. Use of a 3d CAD/modelling program would be more appropriate for rendering and manipulating surfaces. Operations that are extremely simple to execute in CAD packages, such as 'trim' and 'extend', were very demanding in terms of being scripted in Processing language.

- **La Rioja, Bodegas Ysios, Laguardia**

1998-2000 // Architect: Santiago Calatrava

Figure 22. Bodegas Ysios (Images from http://www.arcspace.com)

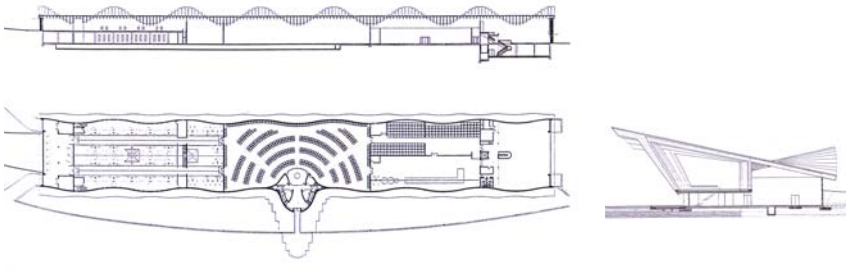


Figure 21. Bodegas Ysios, plan and sections (Images from A+U, 03:03, pp49-59.)

$$\begin{bmatrix} 0 & 5 & 20 & 100 & 0 & 0 & 0 & -50 & 55 & -410 & 0 & PI/2 & 0 \\ 0 & 5 & 20 & 100 & 0 & 0 & 0 & -50 & -55 & -410 & 0 & PI/2 & 0 \\ 0 & 0 & 20 & 10 & 0 & 0 & 0 & 50 & 410 & -55 & PI/2 & 0 & PI/2 \\ 0 & 0 & 20 & 10 & 0 & 0 & 0 & 50 & -410 & -55 & PI/2 & 0 & PI/2 \\ 0 & 40 & 20 & 150 & 0 & 0 & 0 & -70 & 50 & -25 & 0 & PI/2 & -PI/8 \\ 0 & 0 & 21 & 20 & 0 & 6 & 0 & 0 & 90 & -425 & PI/2 & PI/2 & 0 \end{bmatrix}$$

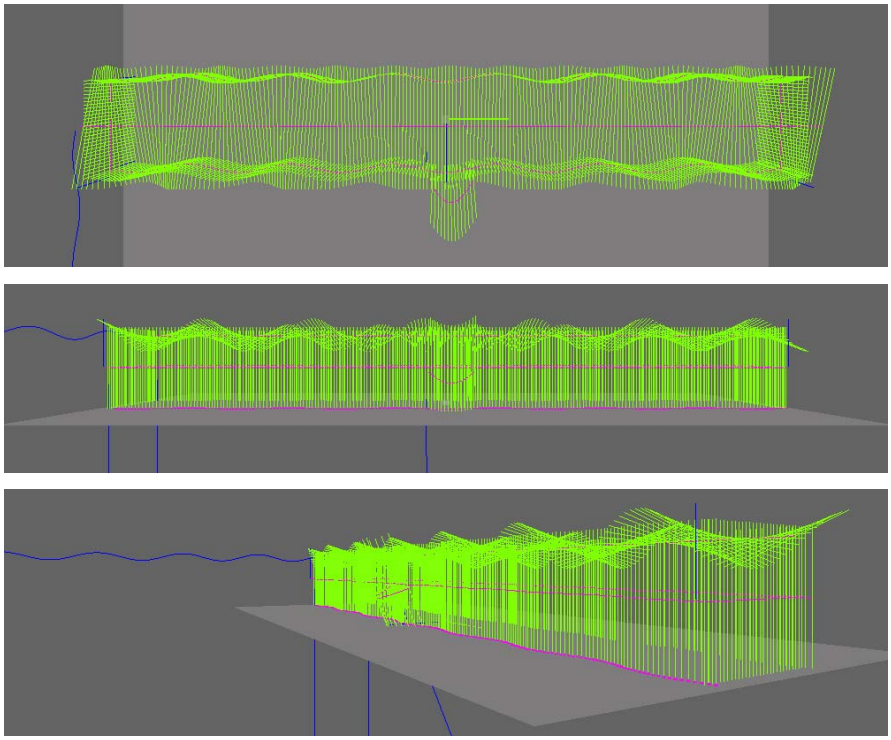


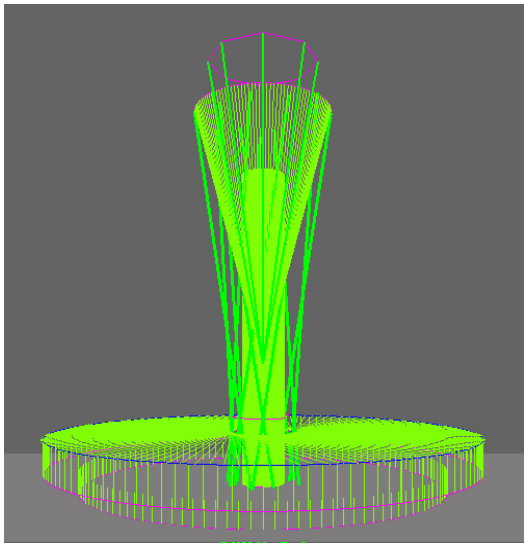
Figure 23. Bodegas Ysios in Processing

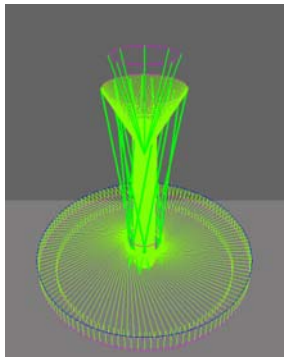
- **Traffic air control tower, El Prat Airport, Barcelona**

2004 // Architect: Ricardo Bofill

Figure 24. Traffic air control tower (Image from http://www.gop.es/AEROPUERTOS/TWR_BARNA)

$$\begin{bmatrix} 170 & 170 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 200 & 200 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 20 & 0 & 0 & 0 \\ 65 & 65 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 360 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 200 & 200 & 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 20 & 20 & 0 & 0 & 30 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & 6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & -6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 53 & 53 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 410 & 0 & 0 & 0 \end{bmatrix}$$





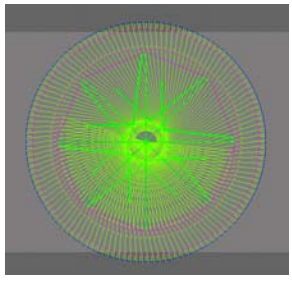
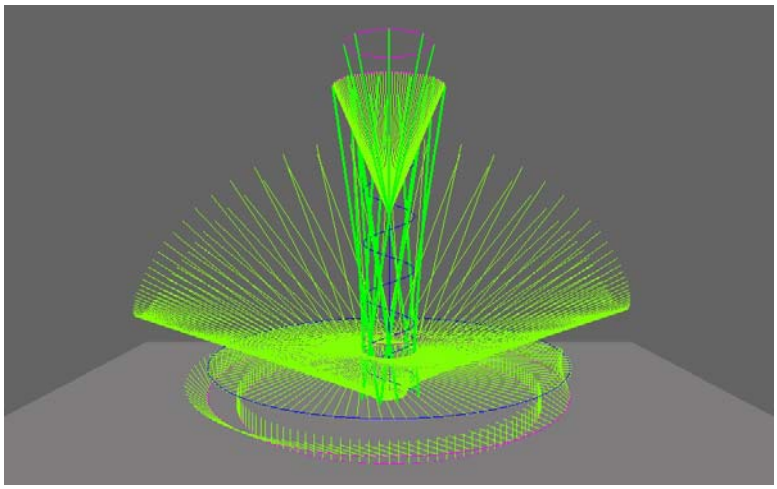
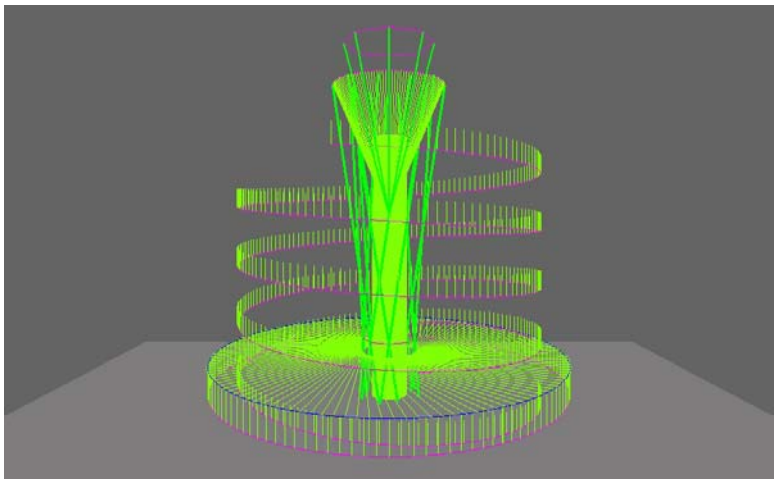


Figure 25. Traffic air control tower in Processing



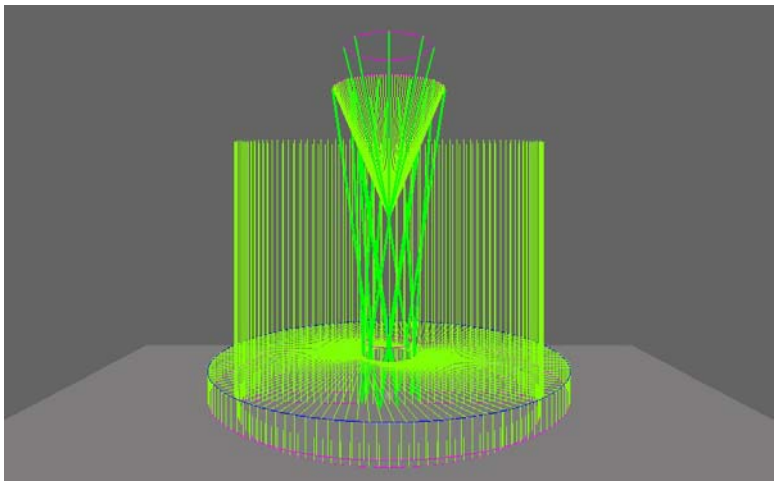
30

$$\begin{bmatrix} 170 & 170 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 200 & 200 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 20 & 0 & 0 & 0 \\ 65 & 65 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 360 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 200 & 200 & 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 20 & 20 & 0 & 0 & 30 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & 6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & -6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 53 & 53 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 410 & 0 & 0 & 0 \end{bmatrix}$$



10

$$\begin{bmatrix} 170 & 170 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 200 & 200 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 20 & 0 & 0 & 0 \\ 65 & 65 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 360 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 200 & 200 & 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 20 & 20 & 0 & 0 & 30 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & 6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & -6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 53 & 53 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 410 & 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 170 & 170 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 200 & 200 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 20 & 0 & 0 & 0 \\ 65 & 65 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 360 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 200 & 200 & 0 & 0 & 0 & 50 & 0 & 0 & 0 \\ 20 & 20 & 0 & 0 & 30 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & 6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 30 & 30 & 0 & 0 & 1 & 1 & -6 & 0 & 0 & 65 & 0 & 0 & 0 \\ 53 & 53 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 410 & 0 & 0 & 0 \end{bmatrix}$$

180

Figure 26. Modifying values...

- **Palace of Assembly, Chandigarh, India**

1953-1963 // Architect: Le Corbusier

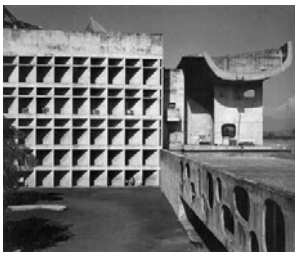


Figure 27. (Image from http://edu.saline.free.fr/01-cites/chandigarh.html)

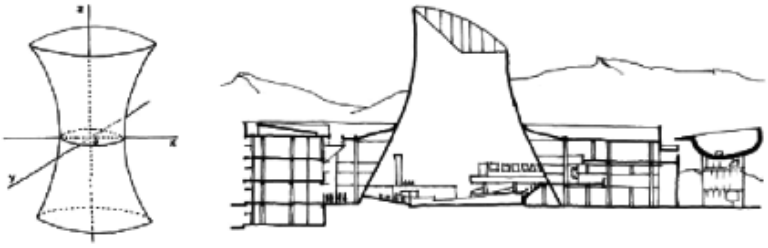


Figure 28. Palace of Assembly, section (Image from http://en.wikipedia.org/)

$$\begin{bmatrix}
45 & 45 & 0 & 0 & 2 & 2 & 5 & 0 & 0 & 180 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & 170 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & -170 & 0 & 0 & 0 & PI/2 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & -170 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & 170 & 0 & 0 & 0 & PI/2 \\
170 & 0 & 0 & 0 & 0 & 0 & 40 & 0 & 157 & 0 & PI/2 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & 145 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & 95 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & 45 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & -5 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & -55 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & -105 & 0 & 0 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & -240 & -155 & 0 & 0 & 0 & 0 \\
30 & 20 & 0 & 0 & 0 & 0 & 10 & -240 & -100 & 0 & -PI/2 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & -120 & 110 & 0 & -PI/6 & PI/2 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & -120 & 110 & -PI/6 & 0 & 0 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & 120 & 110 & 0 & PI/6 & PI/2 \\
0 & 0 & 20 & 10 & 0 & 0 & 0 & 0 & 120 & 110 & PI/6 & 0 & 0
\end{bmatrix}$$

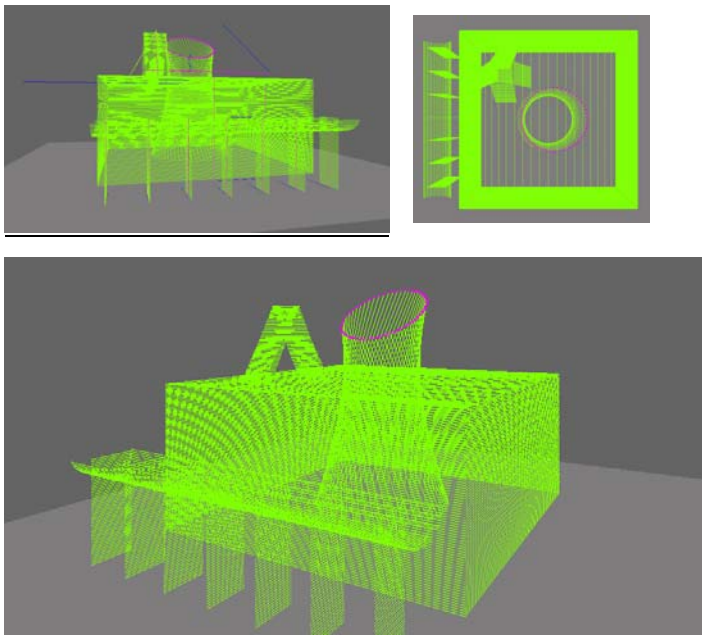


Figure 29. Palace of Assembly in Processing

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

1958 // Architect: Felix Candela

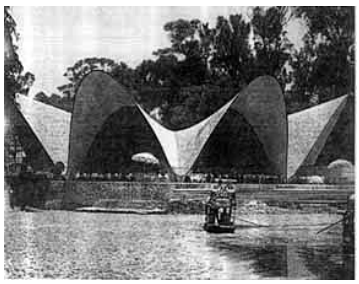


Figure 30. Los Manantiales (R. Bradshaw)

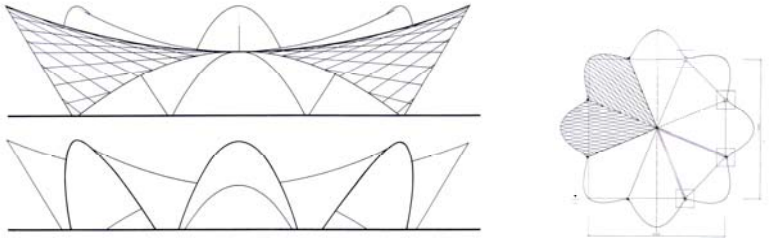


Figure 31. Los Manantiales, elevation and plan (N. Miwa)

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

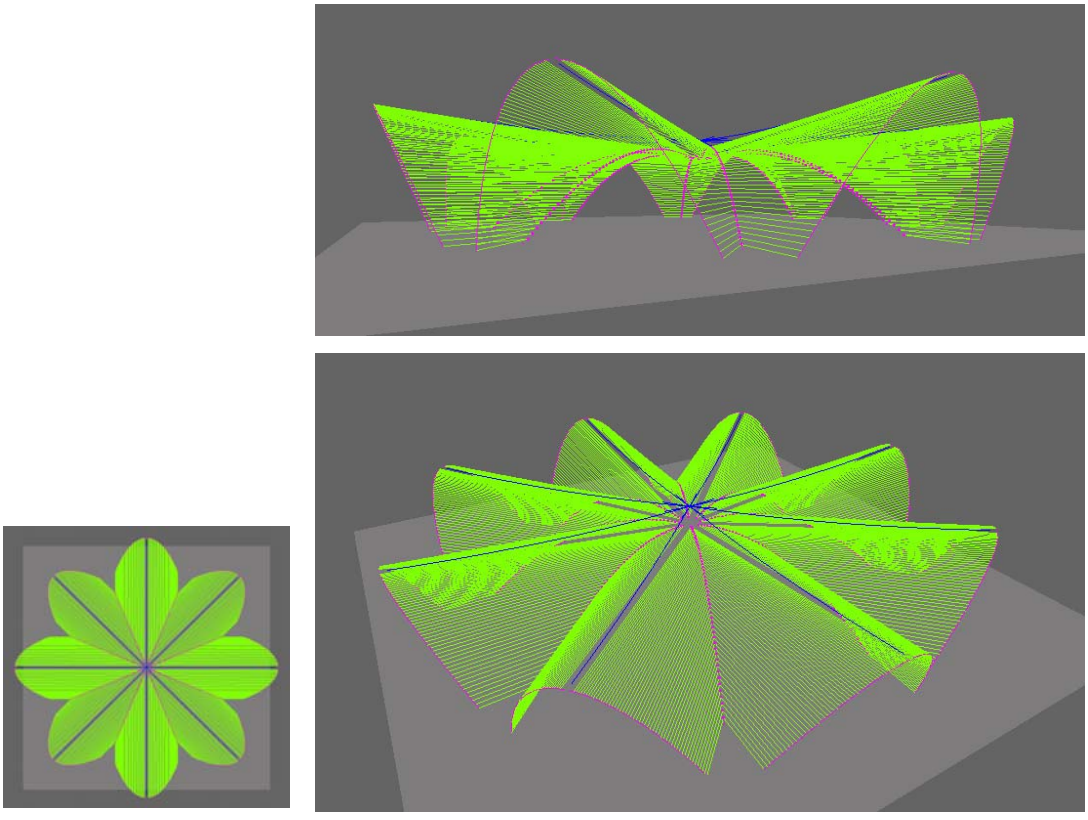


Figure 32: Los Manantiales in Processing.

- **Seagram Building, New York, USA,**

1958 // Architect: Mies Van der Rohe

$$\begin{bmatrix}
40 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\
40 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 40 & 0 & 0 & 0 & 0 \\
18 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 56 & 0 & 0 & 0 & 0 \\
44 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 56 & 0 & 0 & 0 & 0 \\
44 & 0 & 0 & 0 & 0 & 0 & 10 & -25 & 84 & 0 & 0 & 0 & 0 \\
44 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 84 & 0 & 0 & 0 & 0 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 20 & 39.8 & 0 & 0 & 0 & PI/2 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 20 & -39.8 & 0 & 0 & 0 & PI/2 \\
10 & 0 & 0 & 0 & 0 & 0 & 10 & 46 & 17.91 & 0 & 0 & 0 & PI/2 \\
10 & 0 & 0 & 0 & 0 & 0 & 10 & 46 & -17.91 & 0 & 0 & 0 & PI/2 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 64 & 17.91 & 0 & 0 & 0 & PI/2 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 64 & -17.91 & 0 & 0 & 0 & PI/2 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 64 & 43.79 & 0 & 0 & 0 & PI/2 \\
20 & 0 & 0 & 0 & 0 & 0 & 10 & 64 & -43.79 & 0 & 0 & 0 & PI/2
\end{bmatrix}$$

Figure 33: Seagram building (Keller) …

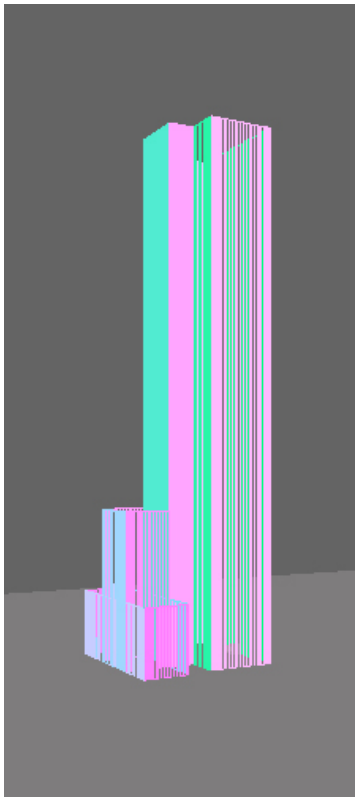


Figure 34: In Processing

5. Discussion

Although the represented buildings were not too complex, they were composed of various types of surfaces and therefore considered appropriate for testing. The selected structures did not cover the full range of building types; that was not feasible due to the number of building types and the time constraint. It also has to be mentioned that existing buildings have been constructed so far based on a very limited number of forms. Thus, the results were limited to the conventional forms and could not expose the system's broad potential. However, it was possible to make a general appraisal of the system, the range of building types it can represent and its limitations.

Comparing to other approaches

Useful remarks were made in relation to other approaches.
The number of values required to represent the system can be quite large, in comparison to March's compact representation which requires a hexadecimal number and dimensioning sets. This is mainly because qualitative aspects of configuration are separated from quantitative in his approach. On the other hand, the parametric system is capable of presenting a significant amount of information in the matrix of the values. Looking at the matrix, information can be instantly acquired about how many types of surfaces compose the structure, what type of surfaces they are -planar or curved-, various height levels, existence or not of repeated elements, relation between elements' location etc. Also same values, possibly signifying common features, can be read at a glance. All this type of information is missing from March's model. What this system lacks is the ability to assign a unique number to each building.

Steadman's archetypal building also has the advantage of being a generic and dimensionless configuration but as such it is quite inflexible; form is shaped only as a stack of storeys of equal height. Dimensions are parameterised and can be transformed but in a very restricted way: only storey heights and depths in strips of accommodation. The parametric system cannot be considered as dimensionless since it requires values for the form to take shape, but there are more possibilities in the underlying relations of the elements.

Both models of Steadman and March are limited only to rectangular forms. Besides that the binary encoding indicates that the elements comprising the form –cubelets or strips or cubes- are either there or not there. Moreover the underlying relations are relations of proximity and vertical-horizontal packing. The parametric system is flexible enough to represent numerous variations of form with elements related without being necessarily physically connected.

It is interesting to note that although the two approaches use binary encoding, they are not suitable for form generation by application in a genetic algorithm. March's cubes can either be suppressed or not, which could result in many unfeasible random solutions, with cubes floating over empty space, for example. Steadman's approach that can include only two shapes of floor plans cannot yield anything really unexpected.

As presented in the literature review, representations of evolutionary design that have to do with architecture tend to combine genetic algorithms with L-systems or shape grammars, the former as the control mechanism of the latter (Kalay 2004). Genes represent design actions as production rules or drawing instructions that when executed result in shapes.

These models are restricted to very simple rules, according to which relations can only exist between one element and the next. An extensive context of interrelations is missing. Additionally, forms generated in proposed approaches are too simple, either two-dimensional plans of three-dimensional simple primitives.
Surfaces used in the parametric system can have very complex forms. By storing information about each individual line, control of the overall configuration is increased. Also they are suitable for inner space division or external skin, in more detail or less detail. Thus, this system can be considered as more efficient for form description covering a broad range of building types.

The approach presented in this thesis is probably closer to Todd and Latham's Evolutionary Art, where relation between parts already exist and parameters are evolved taking random values. Same as in Bentley's approach, use of constructive geometry seems promising. However it is probably too complicated to be integrated into architectural design.

General characteristics of the system

By expressing a structure as a series of values that correspond to a special configuration of elements, the designer can instantly create multiple variations of form presented as three-dimensional models. He can manipulate the models on the screen rather than simply imagine how they would look like.
Having the relations of surfaces specified in the system and specific values defined by the user incorporates in the system a kind of interactivity. The user interacts by altering dimensions and the model responds to him by being modified. The model can be modified a limitless amount of times while always maintaining relations between parts of a volume during modifications. Embedding rules and constrains in the representation, in the form

of parameters transforms the otherwise "passive" representation into an active "one" (Kalay 2004).

It has been observed from the results that small changes in values can lead to unexpected changes of form. Thus experimenting with random values can lead to interesting creations. Creativity can be enhanced.

Equally important is that identification of the relations makes the final product comprehensible from a structural point of view. The way a change in on element influences the other becomes more comprehensible.

The use of a very small number of variables in the present system renders it highly flexible. Elements can be assembled in an unlimited number of ways. Representation of almost all forms and consequently of almost all building types can, in theory, be achieved.

Other characteristics of the system that should be mentioned are its efficiency in terms of speed and simplicity. Embedding rules and relations between elements of the system makes it very easy to understand, manipulate and use. Without special effort a three-dimensional model can be obtained simply by entering a number of values. The process is very fast affirming that design and modification of sets of objects that have already parametric rules embedded in them can be faster (Kalay 2004).

Limitations

Some forms are too complex to be expressed by this system of representation. It is not applicable to buildings that are composed of ruled surfaces swept along a free form curve, such as DeCOi's Paramorph. Apparently, neither is it applicable to buildings composed of free form surfaces, such as the Guggenheim Museum in Bilbao. Free form surfaces too complex to be generalised and

described as a restricted set of parameters, when the intention is to keep the number of variables low.

Another possible drawback is that a proper way for controlling variants that might produce not valid results has not been integrated in the system.

Application in a Genetic Algorithm

What remains unanswered is the results this representation would produce if used in an evolutionary search procedure with the intention of form generation; the integration of the representation in a Genetic Algorithm, for example.

Peter Bentley (Bentley, 1999) explains that Genetic Algorithms maintain a population of individuals where each individual consist of a genotype and a corresponding phenotype. Phenotypes are coded solutions to the design problem and usually consist of collections of parameters. Genotypes consist of coded versions of these parameters. A coded parameter is normally referred to as gene, with the values a gene can take being known as alleles. Collections of genes often organised as strings. Genotypes must be mapped onto phenotypes –the actual solutions - before the quality or fitness of each solution can be evaluated.

If the proposed representation was used in a Genetic Algorithm, the parameters that describe surface would be mapped into a genotype, whose genes would correspond to the set of values. Parameters would be structured in genes according to their signification. A gene would be structured to hold all the base curve parameters, for example, and another all the director curve parameters. Transformation parameters would be organised in an analogous manner.

An initial set of rules and constraints responding to a given problem and specification of fitness criteria would be required. A fitness test could possibly evaluate how close the values are to a given example of structure. Then populations of alternative solutions would be generated and gradual changes over generations would eventually provide the desirable architectural forms for the specific problem.

Further Work

The next expected step in the system's development would be its integration with 3D CAD/Modeling software. Testing in a programming language like Processing proved to be especially restricting in terms of drawing and rendering. Use of an appropriate program is compulsory in order to produce graphic representations that brings out the system's potential.

A possible improvement to the system of representation would have to do with repeated surfaces. In the resulting matrices it was observed when the same surface is repeated in the building's composition, translated or rotated, it occupies large space in matrix, without providing useful information. This could probably be avoided if a shorter way to express repeated surfaces was defined.

A more general development could be the implementation of constraints that connect one surface to the other. Parallelism, perpendicularity, tangency, dimensionality are such constraints. Constraints can also be specified as conditional relations. Incorporating conditional expressions might extend the interest of the method.

The way to incorporate free form curves, could also be investigated -probably only as the base curve to keep the number of parameters low-.

Conclusions

Architectural structures are far too complex to be represented accurately. Each system should focus on certain aspects of the structure and certain stages of the design process. Different representations are required for specific purposes. Conventional techniques usually involved making decisions of what aspects of design to represent or not, so that the designer can focus on specific features at each stage of the process. The thesis is considering implications that the input/role of representation in the design process may be extended.

The focus of this thesis has been the representation of architectural structures in an innovative and efficient way, so that it satisfies the digital design demands and further enhances the design process.

It has been justified that the inherent qualities of ruled surfaces render them suitable to comprise the basic elements of an associative representation. A simple parametric system was set up which represents form as an assembly of ruled surfaces, each defined by twelve parameters. Having a limited number of variables increased flexibility of representation.

The most crucial point in the process was the specification of certain relations between elements that are both simple and economic; in this case the variations produced by the degeneration of the helix.

Form defined based on interconnections of elements allows an interactive manipulation of the model by changing parameters. Multiple variations of the model are rapidly produced so that design is converted to selection of the optimal variation. The procedure is very fast, simple to use and can provide unexpected

variations. Significant amount of information about the building is included in the matrix of values.

The rich spectrum of forms makes the representation of almost all architectural forms possible, including planar and a large number of curvilinear forms. The proposed representation is not ideal for conventional building types -that was not the intention anyway-.

Testing was limited to existing buildings so that the system's potential not fully explored. It has been suggested that its integaration in a genetic algorithm might be quite successful. That would constitute the necessary step from form variation to form generation.

Overall, it has been demonstrated how a simple parametric system based on relations of elements and using limited amount of data can be developed as a useful design tool.

Bibliography

1. Achten H., Oxman R, Bax T., 2002, *Typological knowledge acquisition through a schema of generic representations*. Proceedings of Artificial Intelligence in Design'02.

2. Bentley, P., 1999, An Introduction to Evolutionary Design by Computers. In P. Bentley, ed. *Evolutionary design by Computers*, San Francisco: Morgan Kaufmann.

3. Bentley, P., 1999, From Coffee Tables to Hospitals: Generic Evolutionary Design. In P. Bentley, ed. *Evolutionary design by Computers*, San Francisco: Morgan Kaufmann.

4. Bradshaw, R., Campbell, D. , Gargari, M. ,Mirmiran, A. , Tripeny, P, *Special Structures: Past, Present, and Future*, 150th Anniversary Paper, American Society of Civil Engineers.

5. Burry, M., 1993. *Expiatory Church of the Sagrada Família*, London: Phaidon Press.

6. Burry, M., 2003. Between Intuition and Process: Parametric Design and Rapid Prototyping. In B. Kolarevic, ed. *Architecture in the Digital Age- Design and Manufacturing*, New York: Spon Press.

7. Burry, M., Burry J., Dunlop G.M., Maher A., *Drawing Together Euclidean and Topological Threads*, presented at SIRC 2001.

8. Coates, P., Broughton, T., Jackson, H., 1999, *Exploring* Three-Dimensional Design Worlds using Lindenmayer Systems and Genetic Programming. In P. Bentley, ed. *Evolutionary design by Computers*, San Francisco: Morgan Kaufmann.

9. DeCOi, 2000, Technological Latency: from Autoplastic to Alloplastic, *Digital Creativity*, 11 (3), pp.131-143.

10. Emch, A., 1919, *On a Certain Class of Rational Ruled Surfaces*. In Proceedings of the National Academy of Sciences, HighWire Press.

11. Ervin, S., Hasbrouck, H., 2001, *Landscape Modeling: Digital Techniques for Landscape Visualization.* New York: McGraw-Hill.

12. Evans, R., 1995, *The Projective Cast : Architecture and its Three Geometries*. Cambridge: The MIT Press.

13. Frazer, J., 1995, *An Evolutionary Architecture*. London: Architectural Association Publications.

14. Frazer, J., 2002, Creative Design and the Generative Evolutionary Paradigm. In P. Bentley, W. David Corne, eds, *Creative evolutionary systems*, San Francisco: Morgan Kaufmann.

15. Gray, A., 1993, *Modern Differential Geometry of Curves and Surfaces with Mathematica.* FL: CRC Press.

16. Hofstadter, D.R., 1980, *Gödel, Escher, Bach*. London: Penguin Books.

17. Jackson, H., Toward a Symbiotic Coevolutionary Approach to Architecture. In P. Bentley, W. David Corne, eds, *Creative evolutionary systems*, San Francisco: Morgan Kaufmann.

18. Kalay, Y.E., 2004, *Architecture's New Media : Principles, Theories, and Methods of Computer-Aided Design*. Cambridge: MIT Press.

19. Keller, S., Fenlad Tech: *Architectural Science in Postwar Cambridge.* Available from: http://www.mitpressjournals.org/

20. Kolarevic, B., 2003, Digital Morphogenesis. In B. Kolarevic, ed. *Architecture in the Digital Age- Design and Manufacturing*, New York: Spon Press.

21. Krishnapillai, A., *Genometry: a Genetically Inspired Parametric Form Generation Method.* Available from: www.arch. usyd.edu.au/ kcdc/conferences/dcc04/workshops/workshopnotes3

22. Kühnel, W., 2002, *Differential Geometry : Curves - Surfaces – Manifolds.* Rhode Island: AMS.

23. March, L., 1972, *A Boolean description of a class of built forms.* Cambridge: Cambridge University Press.

24. March, L., Steadman, P., 1971, *The Geometry of the Environment*, London: R.I.B.A Publications.

25. Miwa, N., ed. 1995. *Felix Candela*. Tokyo: TOTO Shuppan.

26. O'Neill, B., 1966, *Elementary Differential Geometry.* New York: Academic Press.

27. Steadman, P., 1998, Sketch for an archetypal building. *Environment and Planning B: Planning and Design*, 25th Anniversary Issue, pp.92-105.

28. Steadman, P., 2001, Binary encoding of a Class of Rectangular Built-Forms. *Proceedings*, 3rd International Space Syntax Symposium, Atlanta.

29. Steadman, P., Waddoups, L., 2000, A Catalogue of Built Forms, using a Binary Representation. *Proceedings*, 5th International Conference on Design and Decision Support Systems in Architecture, Nijkerk, The Netherlands pp.353-373.

30. Todd, S., Latham, W., 1999, The Mutation and Growth of Art by Computers. In P. Bentley, ed. *Evolutionary design by Computers*, San Francisco: Morgan Kaufmann.

31. Rosenman, M., Gero, J., 1999, Evolving Designsby Generating Useful Complex Gene Structures. In P. Bentley, ed. *Evolutionary design by Computers*, San Francisco: Morgan Kaufmann.

32. Krishnapillai, A., *Genometry: a Genetically Inspired Parametric Form Generation Method.* Available from: www.arch. usyd.edu.au/ kcdc/conferences/dcc04/workshops/workshopnotes3

Appendix I

"There exists, for example, a one-to-one correspondence between ruled surfaces and a certain class of partial differential equations, so that the theories of the two classes are abstractly identical.

…

A much favored method, especially in descriptive geometry, consists in considering ruled surfaces as continuous sets of straight lines, or generatrices which intersect three fixed curves, the directrices, simultaneously.

…

Frequently, ruled surfaces are also defined as systems of elements, either common to two rectilinear congruences, or to three rectilinear complexes.

…

Of great importance is the definition of ruled surfaces as systems of lines which join corresponding points of an (a,b)-correspondence between the points of two algebraic curves *Cm* and *Cn* of orders *m* and *n.*

…

Finally there is the cinematic method in which ruled surfaces are generated by the continuous movement of the generatrix according to some definite cinematical law."

Appendix II

Code snippet [a]

```
Surface()
  {…
    for (int i=0; i<num; i++)
    {
      t+=2*PI/(2*e);
      base i]=  new Vec(ab*cos(t),bb*sin(t),cb*t);
      dir [i]=  new Vec(ad*t+bd*cos(t),cd*sin(t),dd*t);
    }
  }
  void draw()
  {
    pushMatrix();
    for (int i = 1; i < num; i++) {
      //.............. ............................... draw two curves
      stroke(0,0,255);              // draw director curve
      line(dir[i].m_vec[0],dir[i].m_vec[1],dir[i].m_vec[2],
      dir[i-1].m_vec[0],dir[i-1].m_vec[1],dir[i-1].m_vec[2]);
      stroke(255,0,255);             // draw base curve
      line(base[i].m_vec[0],base[i].m_vec[1],base[i].m_vec[2],
      base[i-1].m_vec[0],base[i-1].m_vec[1],base[i-1].m_vec[2]);
      stroke(0,255,0);
      //.............................................................
      slope[i]= new Vec(dir[i].m_vec[0]-dir[i-1].m_vec[0],dir[i].m_vec[1]-
dir[i-1].m_vec[1],dir[i].m_vec[2]-dir[i-1].m_vec[2]);
      slope[i].normalize();
      lineEnd[i]=new
Vec(base[i].m_vec[0]+slope[i].m_vec[0]*len,base[i].m_vec[1]+s
lope[i].m_vec[1]*len,base[i].m_vec[2]+slope[i].m_vec[2]*len);
     line (base[i].m_vec[0],base[i].m_vec[1],base[i].m_vec[2],
lineEnd[i].m_vec[0],lineEnd[i].m_vec[1],lineEnd[i].m_vec[2]);
    }
     popMatrix();
  }
}
```

Code snippet [b]

```
class Surface
{...
  for (int i=0; i<num; i++)
    {
      t+=2*PI/(2*e);
      baseTemp [i]=  new Vec(ab*cos(t),bb*sin(t),cb*t);
      baseTemp1[i]= trans (baseTemp[i], transX, transY, transZ);
      baseTemp2[i]= rotX (baseTemp1[i], rotX);
      baseTemp3[i]= rotY (baseTemp2[i], rotY);
      base[i]= rotZ (baseTemp3[i], rotZ);

      dirTemp [i]=  new Vec(ad*t+bd*cos(t),cd*sin(t),dd*t);
      dirTemp1[i]= trans (dirTemp[i], transX, transY, transZ);
      dirTemp2[i]= rotX (dirTemp1[i], rotX);
      dirTemp3[i]= rotY (dirTemp2[i], rotY);
      dir[i]= rotZ (dirTemp3[i], rotZ);
    }
  }



class Vec
…
Vec trans (Vec a, float tx,float ty, float tz)
{
  Vec c= new Vec();
  c.m_vec[0] = a.m_vec[0] + tx;
  c.m_vec[1] = a.m_vec[1] + ty;
  c.m_vec[2] = a.m_vec[2] + tz;
return c;
}
Vec rotX (Vec a, float theta)
{
  Vec c= new Vec();
  c.m_vec[0] = a.m_vec[0];
  c.m_vec[1] = a.m_vec[1] * cos(theta)- a.m_vec[2]*sin(theta);
  c.m_vec[2] = a.m_vec[1] * sin(theta)+ a.m_vec[2]*cos(theta);
return c;
}
Vec rotY (Vec a, float theta)
{
  Vec c= new Vec();
  c.m_vec[0] = a.m_vec[0] * cos(theta)+a.m_vec[2]*sin(theta);
  c.m_vec[1] = a.m_vec[1] ;
  c.m_vec[2] = -a.m_vec[0] * sin(theta)+a.m_vec[2]*cos(theta);
return c;
}
```

Code snippet [c]

```
void setup()
{
  …
  numberOfSurfaces=14;
  s= new Surface [numberOfSurfaces];

  s[5]= new Surface (50, 44, 0,0, 0,0,0,10,   40,  0,84,0,    0,0,0 ) ;
  s[6]= new Surface (50, 20, 0,0, 0,0,0,10,   320,
s[1].base[0].m_vec[1]/2,s[1].base[0].m_vec[0],0 ,      0,0,PI/2) ;
  s[7]= new Surface (50, 20, 0,0, 0,0,0,10,   320, s[1].base[0].m_vec[1]/2,-
s[1].base[0].m_vec[0],0 ,     0,0,PI/2) ;
  s[8]= new Surface (25, 10, 0,0, 0,0,0,10,   320,
18+s[2].base[0].m_vec[1]/2,s[2].base[0].m_vec[0],0 ,      0,0,PI/2) ;
  s[9]= new Surface (25, 10, 0,0, 0,0,0,10,   320,
18+s[2].base[0].m_vec[1]/2,-s[2].base[0].m_vec[0],0 ,     0,0,PI/2) ;
  s[10]= new Surface (20, 20, 0,0, 0,0,0,10,   92,
22+s[4].base[0].m_vec[1]/2,s[2].base[0].m_vec[0],0 ,      0,0,PI/2) ;
  s[11]= new Surface (20, 20, 0,0, 0,0,0,10,   92,
22+s[4].base[0].m_vec[1]/2,-s[2].base[0].m_vec[0],0 ,     0,0,PI/2) ;
  s[12]= new Surface (20, 20, 0,0, 0,0,0,10,   40,
22+s[4].base[0].m_vec[1]/2,s[5].base[0].m_vec[0],0 ,      0,0,PI/2) ;
  s[13]= new Surface (20, 20, 0,0, 0,0,0,10,   40,
22+s[4].base[0].m_vec[1]/2,-s[5].base[0].m_vec[0],0 ,     0,0,PI/2) ;
}
```

Code snippet [d]

```
Vec inter (Vec P3, Vec P1, Vec P2 , Vec N)
//  P3:s1.base[i-1], P1:s2.base[i], P2:s2.lineEnd[i],  N:s1.n
{
  Vec e=new Vec();
  Vec sub1= sub(P3, P1);
  Vec sub2= sub( P2 ,  P1);
  float dot1= dot(N, sub1);
  float dot2= dot( N, sub2);
  float u= dot1/dot2;
  e.m_vec[0] = P1.m_vec[0] + u*(P2 .m_vec[0]-P1.m_vec[0]);
  e.m_vec[1] = P1.m_vec[1] + u*(P2 .m_vec[1]-P1.m_vec[1]);
  e.m_vec[2] = P1.m_vec[2] + u*(P2 .m_vec[2]-P1.m_vec[2]);
  return e;
}


…
void draw()
{…
for (int j=0;j<numberOfSurfaces; j++)
  {
    for (int i=2; i< s[7].num; i++)
    {
 Vec intPoint= inter(s[j+1].base[i-1], s[j].base[i], s[j].lineEnd[i],
s[j+1].n[i]);
      pushMatrix();
      translate(intPoint.m_vec[0],intPoint.m_vec[1],intPoint.m_vec[2]);
      stroke(255,0,255);
      sphere (2);
      popMatrix();
      stroke(128,255,6);
      s[j].lineEnd[i].m_vec[0]=intPoint.m_vec[0];
      s[j].lineEnd[i].m_vec[1]=intPoint.m_vec[1];
      s[j].lineEnd[i].m_vec[2]=intPoint.m_vec[2];
      line (s[j].base[i].m_vec[0],s[j].base[i].m_vec[1],
s[j].base[i].m_vec[2],s[j].lineEnd[i].m_vec[0],s[j].lineEnd[i].m_vec[1],
s[j].lineEnd[i].m_vec[2]);
      s[j+3].lineEnd[i].m_vec[0]=intPoint.m_vec[0];
      s[j+3].lineEnd[i].m_vec[1]=intPoint.m_vec[1];
      s[j+3].lineEnd[i].m_vec[2]=intPoint.m_vec[2];
      line (s[j+3].base[i].m_vec[0],s[j+3].base[i].m_vec[1],
s[j+1].base[i].m_vec[2],s[j].lineEnd[i].m_vec[0],s[j+1].lineEnd[i].m_vec[1],
s[j+1].lineEnd[i].m_vec[2]);
    }
  }
```