Parametric Design: An Implementation of Bentley Systems Generative Components

by
Mark Andrew Cichy

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Architecture
in
Architecture

Waterloo, Ontario, Canada, 2006

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including my required final revisions, as accepted by examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis addresses the need for flexible parametric design tools. It focuses on the implementation of a particular tool, Bentley Systems' Generative Components, by exploring features, strengths and weaknesses, and how features can be implemented in design. An exposition of Generative Components is introduced to bridge the gap between the potential and existing power of parametric tools. Through a case study of the Bahá'í Temple for South America this thesis explores the implementation of Generative Components. The exposition argues for the validity of parametric research, specifically its ability to streamline and enhance an architectural design process.

The topic of parametric design is further documented in a survey submitted to researchers and developers in the field of parametric research and design. The purpose of this documentation is to place the progression of parametric tools within the context of current development, initiating an open-ended discussion focusing on future research.

This thesis adds to the current development of parametric technology by making particular contributions to tools within the realm of parametric research. Primary contributions include array seeking scripts that search for and replace or duplicate objects, routines for nesting functions within scripts, ideological workflow development and conceptual training through practical application.

Acknowledgements

Many thanks to the practice of Hariri Pontarini Architects (HPA) for providing me with access to all of the latest design files since August 2004. The design team at HPA has been incredibly flexible and eager to contribute to this work.

I would like to acknowledge the contributions made by those individuals—both professional and academic—that took part in the rather lengthy query that frames the body of this work; including, Dr. Robert Aish, Lars Hesselgren, Axel Kilian, Hugh Whitehead, Dr. Chris J.K. Williams and Dr. Robert Woodbury.

I must give thanks to the members of my committee—Philip Beesley, Donald McKay and Thomas Seebohm—your patience was highly valued and your comments highly regarded. Philip, you inspire me to go beyond my greatest efforts. Donald, you demand the utmost sense of coherence from my words and thoughts. Thomas, the honesty of the technical aspect of the work is a tribute to your systematic clarity.

A special thanks is given to Dr. Robert Aish and Dr. Robert Woodbury. Their continued support of both my thesis and extra curricular efforts enriched the content of this thesis. The ideologies behind the development of Dr. Aish's work on Generative Components has had a profound impact on my thesis and life long pursuit in the area of parametric design technologies. Their inclusion of my work in publications and research has not gone unnoticed and I thank them for the commendations.

Thank you to my beautiful wife for her unwavering confidence and support. I could not have survived the last two years without her standing by my side.

For my intelligent and evermore beautiful wife Natalie.

Contents

A | CONTEXT

CHAPTER TITLE ILLUSTRATION *This array of simple objects was used to test the limits of Generative Components' Array functionality. The illustration is comprised of several nested arrays, the number of arrayed items exceeds one and a half million objects.*

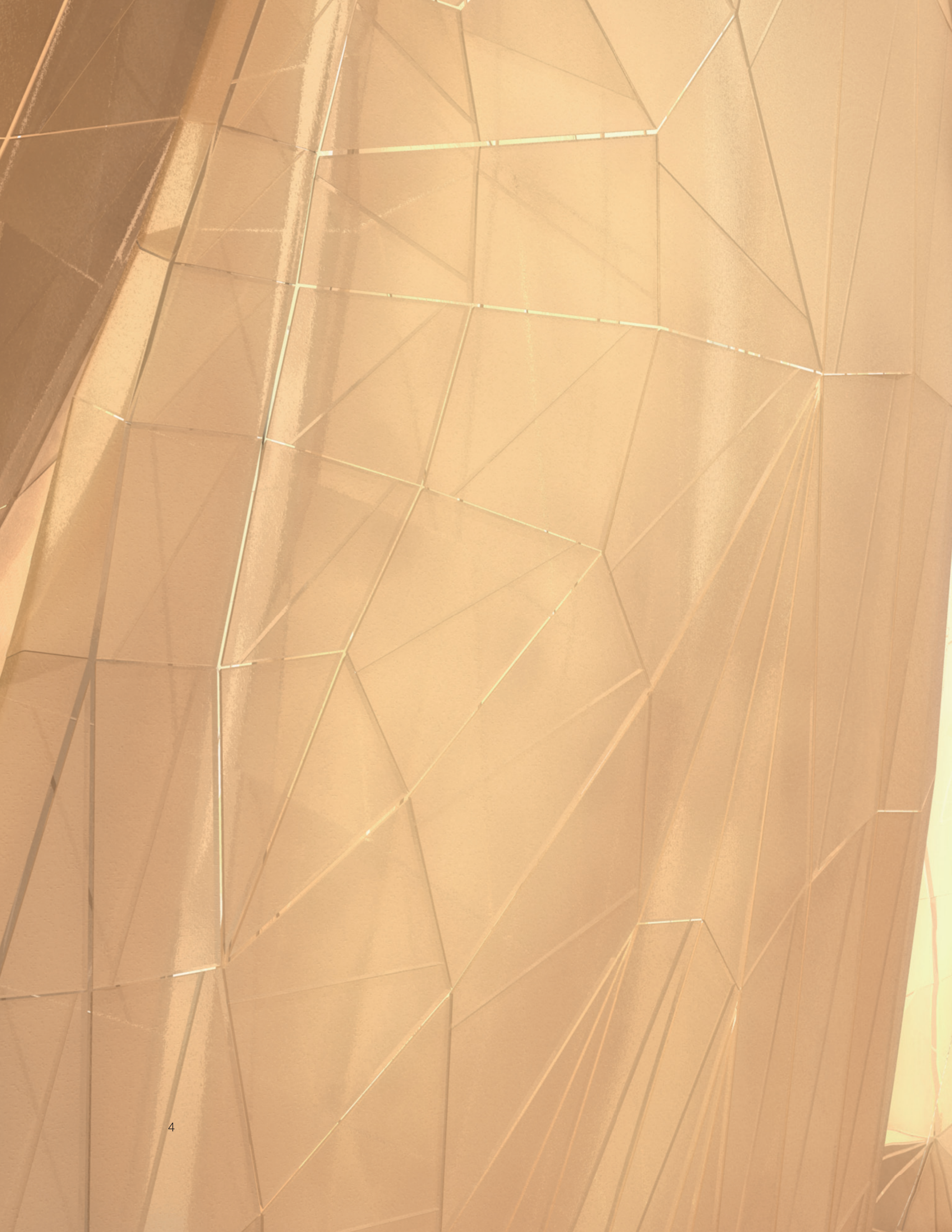
This thesis examines parametric design in architecture by exploring the application of Bentley Systems' Generative Components, using the author's original contributions to the Bahá'í Temple by Hariri Pontarini Architects. These contributions include the development of array seeking scripts, routines for nesting *Functions* within scripts, ideological workflow development and conceptual training through practical application.

The thesis has three major components. The first chapter contains a case study of the Bahá'í Temple for South America. The second is composed of a graphic exposition of Generative Components. The thesis is concluded with a survey issued to practitioners and developers working in the field of parametric research and design.

The Bahá'í Temple by Hariri Pontarini Architects is used as a case study in this thesis for a number of reasons; its designers employed sophisticated parametric software and its formal aspects are unique in the sense that it has an irregularly shaped exterior. At the Temple's conception I was commissioned to prepare several renderings that captured the effects of the buildings' luminescent skin. The case study introduces the Temple and its formal programme, presents the software tool CATIA and demonstrates some of its features. A critique of CATIA is expressed in a brief analysis that outlines the *workflow* of this product. The case study contributes to the thesis by revealing a type of parametric architectural process. It documents a familiar route and the tasks associated with this route of compiling building documentation.

The exposition of Generative Components is introduced to demonstrate the potential power of parametric tools. It focuses on my original contributions to Generative Components. I focus on applied *modeling methodologies*, *customized software features*, *user interface enhancement*, and the generalized testing and assessment of *pre-release technologies.* The tool establishes a unique idiom not commonly found in software technologies: it includes the ability to graphically create and add customized features back into the application. I argue for the potency of parametric tools in their ability to streamline and enhance the design process.

The concluding segment of this thesis is entitled Framework. Its function is to distill



LEFT *1.01 The following rendering reveals the Temple's luminescent qualities at dawn, all structural components have been removed.*

the content of the case study and expository chapters within the context and state of current research efforts. It also contributes pertinent historical references, and demonstrates how practitioners and researchers have developed their own unique parametric processes. The comments within Framework give this thesis a comparative voice, that juxtaposes various working methodologies. This chapter distills the thesis and exposes possible topics of future development.

The original contributions offered in this thesis are an attempt to enhance the relationship between a user and his tool. Customized software components (aspects of the software that enhance its usability) make it easier for the user to communicate or control ideas. For example, embedding graphical controllers within the modeling interface of the Temple controllers allows a user to visually interact with design components. Equations and variables do the math while the user creates changes to the geometry. The technical aspects of this process are completely transparent to the user. Simple exponential equations and short scripts process the information from the visual inputs to produce a result. It is the production of these scripts and association of these functions that are unique.

The notion that a building can be constructed digitally is no ground-breaking revelation.[1] However I believe few architects and designers are utilizing the full potential of the tools available and even fewer are exploring the development of their own tools for *design space exploration* and manufacturing. The Bahá'í Temple project represents the integration of *parametric design tools* and affordable *manufacturing technologies*. These tools allow Hariri Pontarini to create a flexible test environment as they design architectural details and components.

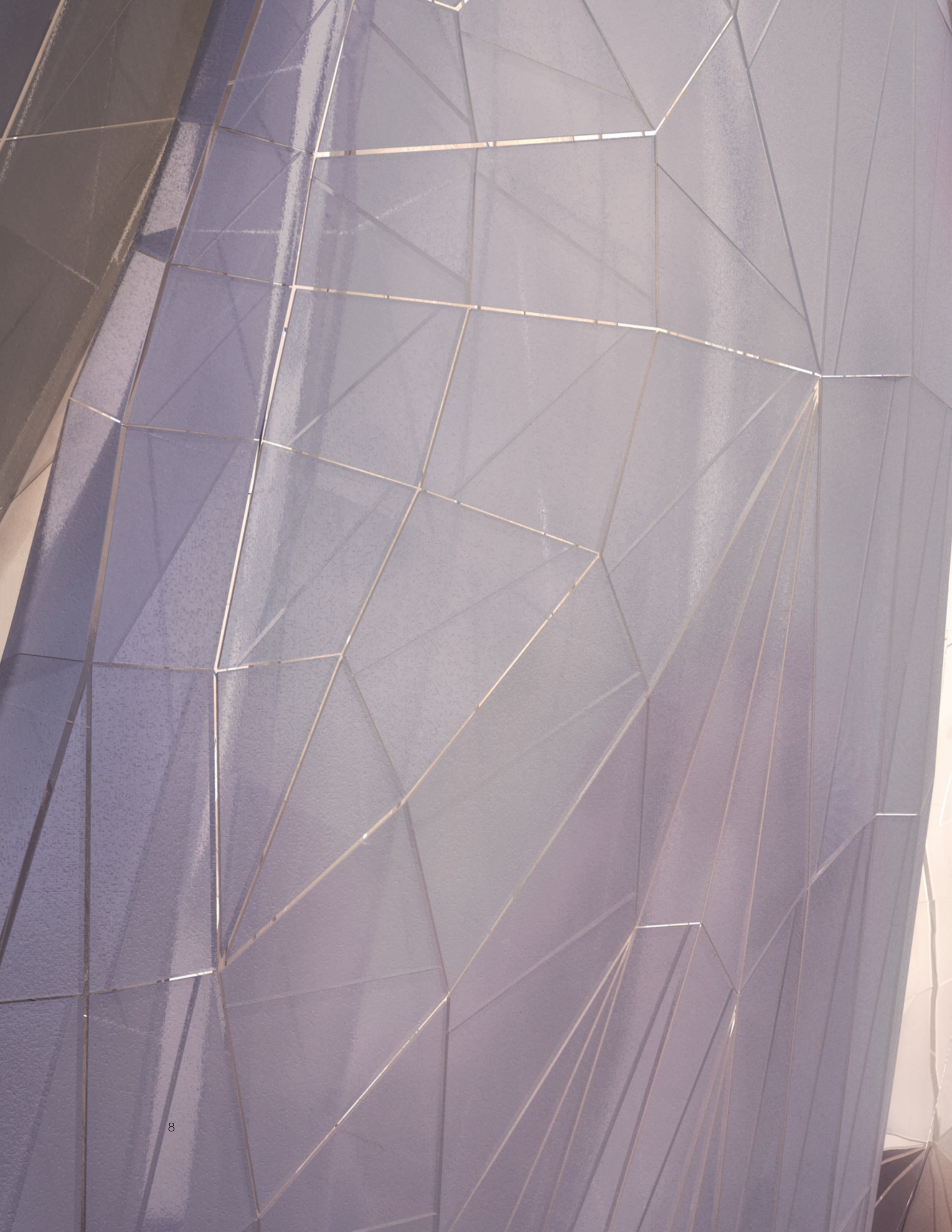
By using parametric modeling tools we are able to explore many design iterations with *greater accuracy*. Parametric relationships are hierarchical and planar, meaning each *component* is capable of redirecting the *system*. Therefore every component has to be *intelligent*, as well as capable of taking control in the event another component fails. Changing the parameters of an *algorithm* or *form* affects each successive element. The interactions are incredibly simple whereas the base code or algorithm is exceedingly complex. Important parameters of the behaviour of the entire set of uncertain systems can be extracted; once the system is built, several values can be extrapolated from instanced *variations* of a form. Embedding this sensibility in three dimensional models

LEFT *1.02 The following rendering reveals the luminescent qualities of the Temple at mid-day. All structural components have been removed.*

and drawings can strengthen the rigour of the architectures' formal concerns.

While parametric tools have existed for decades in engineering disciplines, their availability for architectural practice has only recently emerged. A number of articles and papers have been made available by the introduction of several organizations whose primary focus is designing with digital media. CumInCAD is one such resource. It has had a profound impact on the content of this thesis, primarily in placing the current tools within a larger historical context. "CumInCAD is a cumulative index of publications about computer aided architectural design."[2] Currently parametric tool development is primarily focused on the construction and documentation aspects of engineering and architecture. Several vendors offer parametric products and in some cases a vendor will have more then one product in its *development cycle*. Notable parametric products include Autodesk's Revit and Inventor, Bentley's Generative Components, Dassault Systemes' CATIA and Solidworks, Gehry Technologies' Digital Project (a derivative of CATIA), GraphiSoft's ArchiCAD, and PTC's Pro Engineer. The features of these tools were established to satisfy their respective target industries. ArchiCAD, Digital Project and Revit were designed to suit architectural applications. Their features are primarily focused on streamlining the integration of building components, extrapolating two dimensional drawings and instilling tools to facilitate project management. Inventor, Solidworks and Pro Engineer were designed for engineers and industrial designers. They are primarily used to document "small" products such as power tools, lawn mowers and small to moderately sized mechanical equipment. CATIA was invented for the aircraft, boat building and automobile industries. It is a very *robust* and powerful application and is now being used in certain sophisticated architectural applications. Generative Components (not yet released) cannot be categorized and was developed for architects and designers. Conceptually it can be used by any individual that requires a flexible tool to visually solve design related issues. Although parametric tools have developed into powerful and useful drawing tools, they still cannot be considered effective architectural design tools..

For the purpose of this thesis two parametric design tools have been selected to illustrate how to achieve similar principles in process through differing approaches: CATIA, chosen for its *robust* strengths in modeling and manufacturing; and Generative Components (GC), chosen for its promising openness as a parametric design tool.



LEFT *1.03 The following rendering reveals the Temple's luminescent qualities at dusk, all structural components have been removed.*
NEXT *1.04 Workstation of the CATIA engineer at Hariri Pontarini Architects, the haphazard array of images signifies the drastic amount of process related documentation in constant flux.*

CATIA is an extensible engineering and manufacturing tool that facilitates the production of extremely complex architectural forms.[3] CATIA offers a robust visual interface and propagates change throughout a model by *constraining* objects to one another. Although CATIA facilitates the use of custom *macros* it tends to limit the designer to the context of the tools available within the given software constraints. The interface is extremely sophisticated, its many features are buried within its *Workbenches*. CATIA is a tool that is designed to help facilitate the physical construction and cataloguing of components. CATIA is considered to be a practical tool.

CATIA and Generative Components pursue a different approach to employing the principles of parametric technology. This thesis uses CATIA to document the Fins of the Bahá'í Temple and Generative Components to explore alternative formal variations.

Generative Components (GC) is a flexible design 'toolbox' that allows the creation of *embedded data-driven constructs* for conceptual designers.[4] The *framework* consists of *essential design tools* as well as a highly customizable suite of modeling tools. Linking *Components* generates a high level system that distributes *computationally driven* design problems across an entire *network* of *Sub-Components*. The designer can either rationalize an architectural design through a series of Generative Components, or create form based on the derivation of a generative system of *Components*. GC's inherent bidirectional attributes, and parametric qualities, allow the designer to embed the intent of an architectural design within the construct of an overall system. GC is considered to be a conceptual tool.

Having a single, flexible, and dynamic three dimensional model appeals to most designers. Complex parametric models currently affect performance so greatly that applications become unstable, rendering models useless. Changing factors such as building envelope, square footage and overall form are the most affected by parametric technology. Features of a design that are repetitive, such as windows and doors, are affected by their relation to changing factors. While many parametric tools exist, none are flawless. Establishing a dynamic between static and parametric design components engages the true power of parametric tools.


products
Rhinoceros
NURBS modeling for Windows
IBM
IBM
IBM
IBM
Autodesk VIZ
autodesk
SnowOwl
CATIA
LEGO

B | APPLICATION

COMPOSITION

CHAPTER TITLE ILLUSTRATION *The random array of binary code in this illustration is meant to symbolize the systematic analysis of the chapter. A simple script was created to initialize the first two lines of binary code, these lines were then replicated in a one dimensional array.*

This chapter of the thesis is a case study on the Bahá'í Temple for South America. It establishes the ideologies behind the conceptual design, the context of the Bahá'í faith and contains a purely descriptive assessment of process through the resolution of common architectural design issues.

The chapter is sub-categorized to delineate design issues faced throughout the process. Categories within the Composition component of the chapter, such as Conceptual Sculpting, Organizational Components, Structural Integrity, Axis of Rotation, and Part Detailing describe the distinct dichotomy between design and the post rationalization of components through parametric modeling. Each category outlines an issue and describe a solution.

Skin and Structure describes the relationship between components of the exterior fins; it documents the relationship between cladding materials and space frame super-structure. An orthogonal set of vertical sections (cut along the Z-axis) is the most effective way of documenting this relationship. It is a conventional architectural method of describing such a relationship, and in the case of the Temple, has the distinct advantage of being the most descriptive.

Programmable Analysis addresses certain components of CATIA and how it is used within an architectural process. The content in this section is not meant to be a walkthrough of the application; it is a brief introduction to some of the components of the tool in relation to aspects of the Temple model. As a brief applied study of the tool it provides a quick primer of some of the benefits and weaknesses of CATIA. This sub-section illustrates the progression of the current process of the Temple model.

The CATIA model and plans were supplied by Hariri Pontarini Architects for this study. All other drawings and models, including stress tests, visualization models, and diagrams were created by the author. The translation of parametric CATIA models into static mesh models consisted of complex four or five step processes. The CATIA components were exported piece by piece, normals were corrected, curved surfaces were then meshed and patched, and the components were then imported into static visualization software. Two dimensional drawings, sections and other documentation were directly extracted from within CATIA, exported to third party CAD software and then translated into editable vector graphics.

This chapter provides a base by which the Exposition is founded. It documents the existing process used in the documentation of the Temple and reveals some of the driving concepts of the project. While the content is almost entirely original, for the purpose of staying true to the current process the CATIA model has not been altered. The visualization studies consist of original modeling components, all components presented in these renderings are meant to express and enhance the possible reality of the Temple.

The Bahá'í Faith, established by Bahá'u'lláh

Bahá'u'lláh is an Arabic word, which means "The Glory of God."[1] The religion was established as a vision to re-establish the fundamental definition of human relationships. The faith focuses on human beings themselves, the relationship between human beings and the natural world, the relationship between the individual and society, and the relationship between the members of society and its institutions.

Bahá'u'lláh asserts the deep connection between the practical and spiritual dimensions of human existence. The creation of social structures that promote the development of both individual and collective capacities are of utmost importance.

The nine pointed star is a prominent symbol of the Bahá'í Faith, the significance of the number nine is disputed amongst scholars but is officially defined in the following text; First, it symbolizes the nine great world religions of which we have any definite historical knowledge, including the Babi and Bahá'í Revelations; second, as the highest single digit number it represents the number of perfection; third, it is the numerical value of the word "Baha."[2] The Faith regards humanity as an organic entity which has developed through its embryonic state to infancy, then to adolescence and is now coming of age. The number nine reflects a sense of fulfilment or culmination. All Bahá'í Houses of Worship have nine sides.

While the symbolic use of numbers in Sacred Writings is important, there is no occult meaning to them, nor do Bahá'ís subscribe to divination by numbers.

The Bahá'í Temple for South America

The Bahá'í Temple in Santiago, Chile is designed by Toronto based firm, Hariri Pontarini Architects. The design stands atop a mountain, glowing outwards through a cast glass and alabaster skin. HPA were able to guide the manufacturing of structural components by using Dassault Systemes' CATIA to accurately detail the digital model and provide the *fabricator* with the necessary information to manufacture scale mock-ups. Problems arose during the CATIA modeling process that were not considered at the conceptual phase: problems such as limitations to the facetization of the complex shell structure due to the physical limitations of the materials. Using static tools for the conceptual design process increased the amount of work required on the back end of the project; a considerable amount of design data required reconstruction. Using parametric tools (in this instance, CATIA) required the designer to reconsider the construction of the digital model.

The design intent for The Bahá'í Temple for South America is captured in an excerpt of Siamak Hariri's abstract entitled, "A Temple of Light."[3] It provides a poetic account of the project from concept to structure:

> Light is the fundamental connecting force of the universe. The Temple of Light we have designed employs both translucent stone and the newest glass technology as the means of generating and manifesting both the physiological and spiritual delights of natural light embodied in architecture.
>
> The Temple's nine enfolding wings, identical in form, are organically shaped and twisted slightly to produce, in aggregate, a rather nest-like structure, readable as a soft undulating dome positioned around a raised base. The Temple is to be sited amidst an extensive radiating garden comprising nine reflecting lily pools and nine prayer-gardens.

RIGHT *2.01 The following conceptual rendering of the Temple's skin expresses the translucency Siamak describes in his essay.*

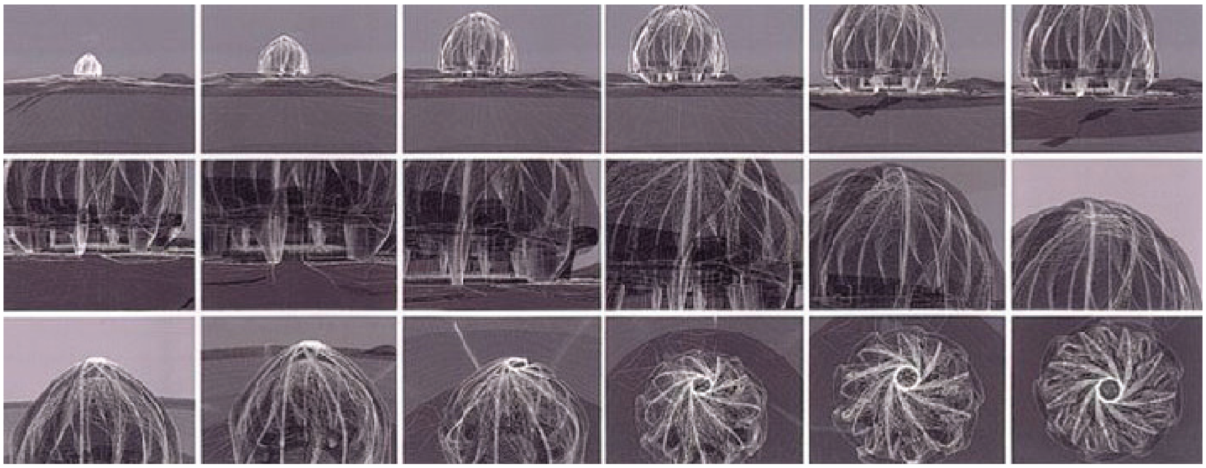
Conceptual Sculpting

The conceptual sketches of the Bahá'í Temple by HPA employed traditional three-dimensional modeling software. Since these products do not typically require a high level of precision it is simple to establish form while dismissing any sense of constructability. This process is sculptural as opposed to buildable.

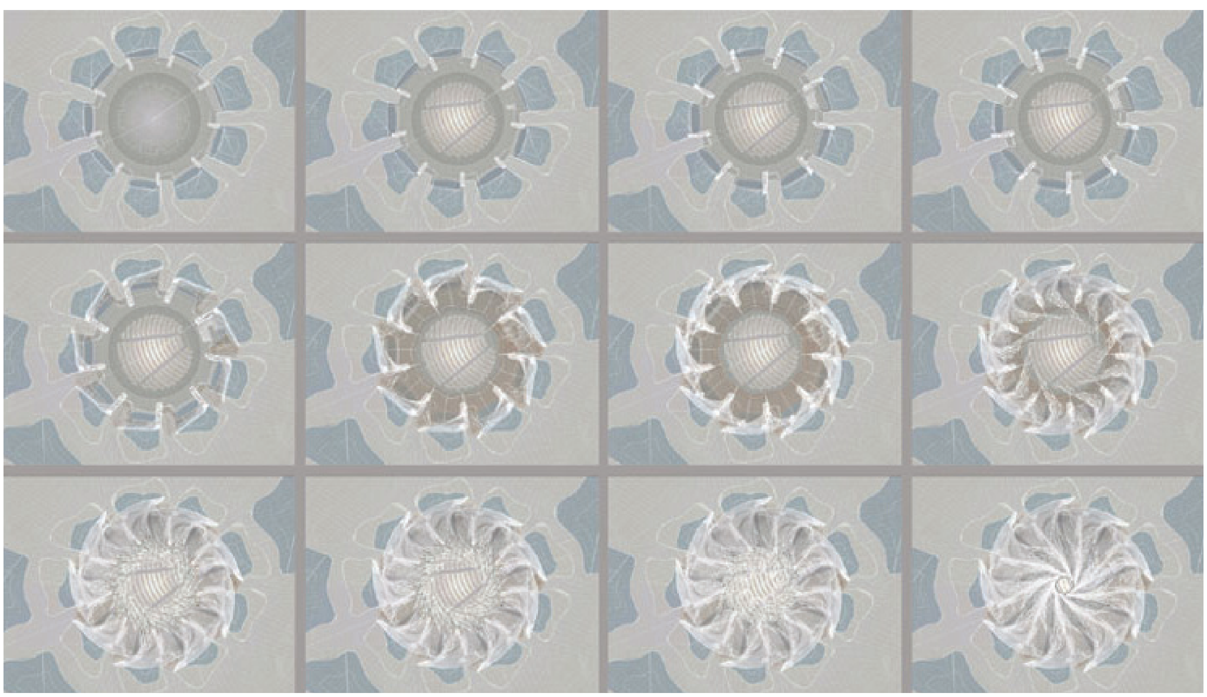
Controlling the amount of extraneous conceptual data strengthens the relationship between concept and construct. Using dedicated manufacturing software only allows for the un-buildable to become buildable and progressively distills concepts as the process advances.

The design for the Bahá'í Temple abides by the guidelines of the traditional design process. It begins with conceptual sketches, preliminary drawings and comprehensive details, which are then used to fabricate the building and its components.

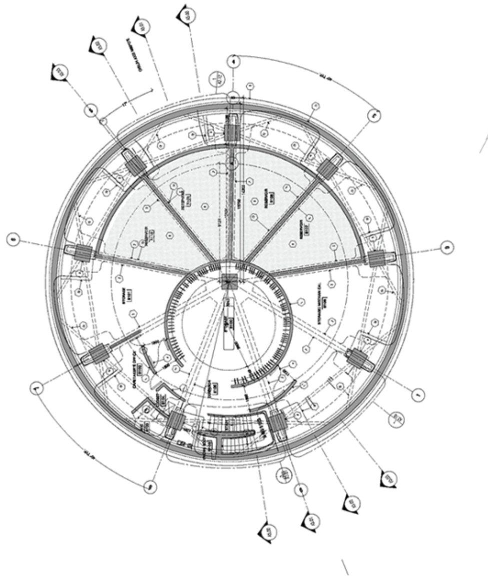
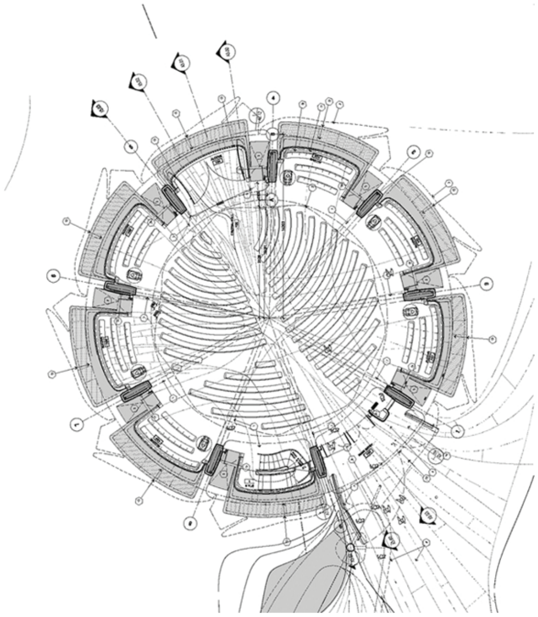
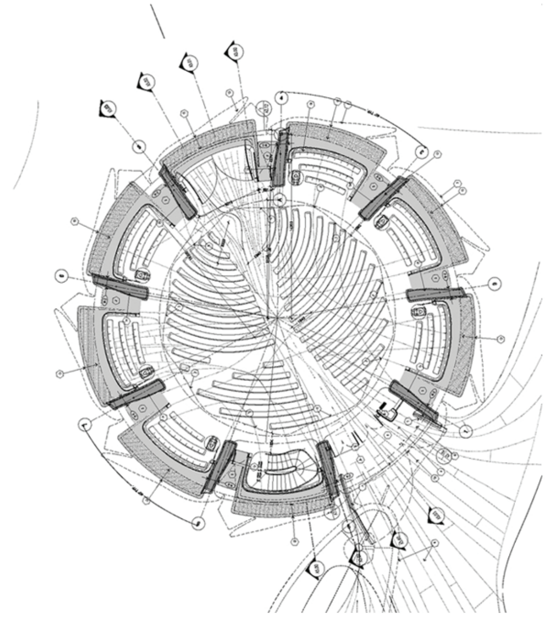
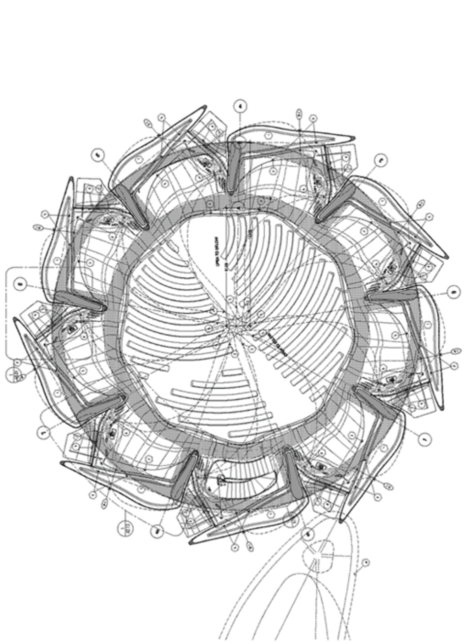
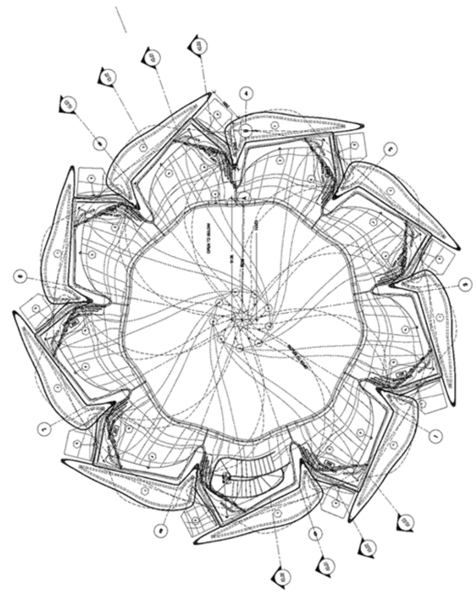
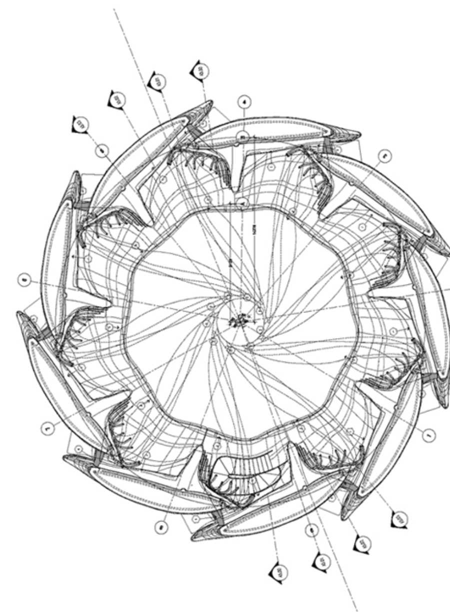
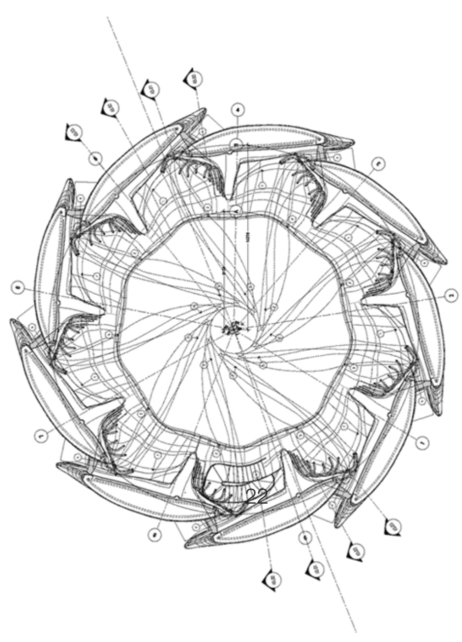
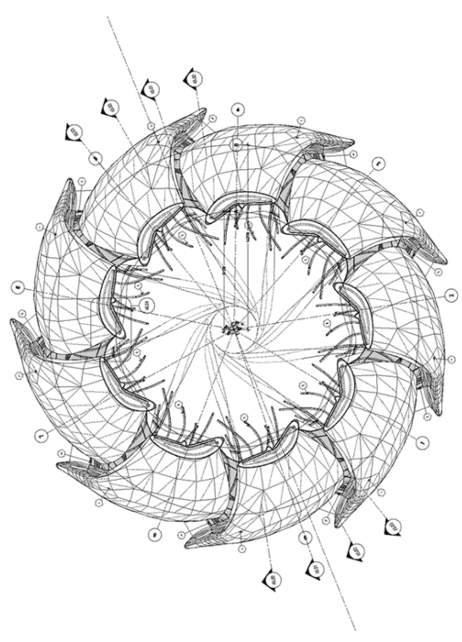
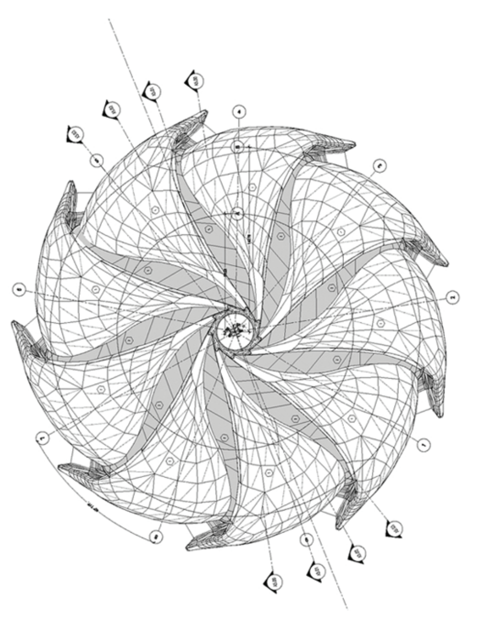
The following description of the Temple employs animated sequences to demonstrate the complexity of the geometry, through visual variations in pattern and form (see 2.02 and 2.03).



2.02 Frames from an early conceptual version of the design



2.03 These sections were completed during the preliminary stages of design

Organizational Parts

Siamak Hariri's "Temple of Light" describes the conceptual principles underlying the formal aspects of the Bahá'í Temple for South America. In the following section of the thesis, we examine the Temple's 'Fin' elements..

The *Parts* of the Fins are modeled as independent components and referenced within the final CATIA *Product*. The Fin Product contains the triangulated alabaster stone, the steel framing, the space frame structure, along with the ball joints, custom cut iron plates, and cast-glass cladding. The Parts are individual CATIA files (.CATPart) that were created as independent components of the Product (.CATProduct). When the Parts are placed inside the Product, they adapt to surrounding Parts through dimensional constraints. The CATIA model's structure is a digital mock-up of the physical Temple. The model is used to test physical variation and document every aspect of the Temple's construction. The hierarchy of the Parts in the CATIA model follows the same hierarchy of the Fins physical building components.

LEFT *2.04 The current floor plans for the Bahá'í Temple express its overwhelming complexities.*

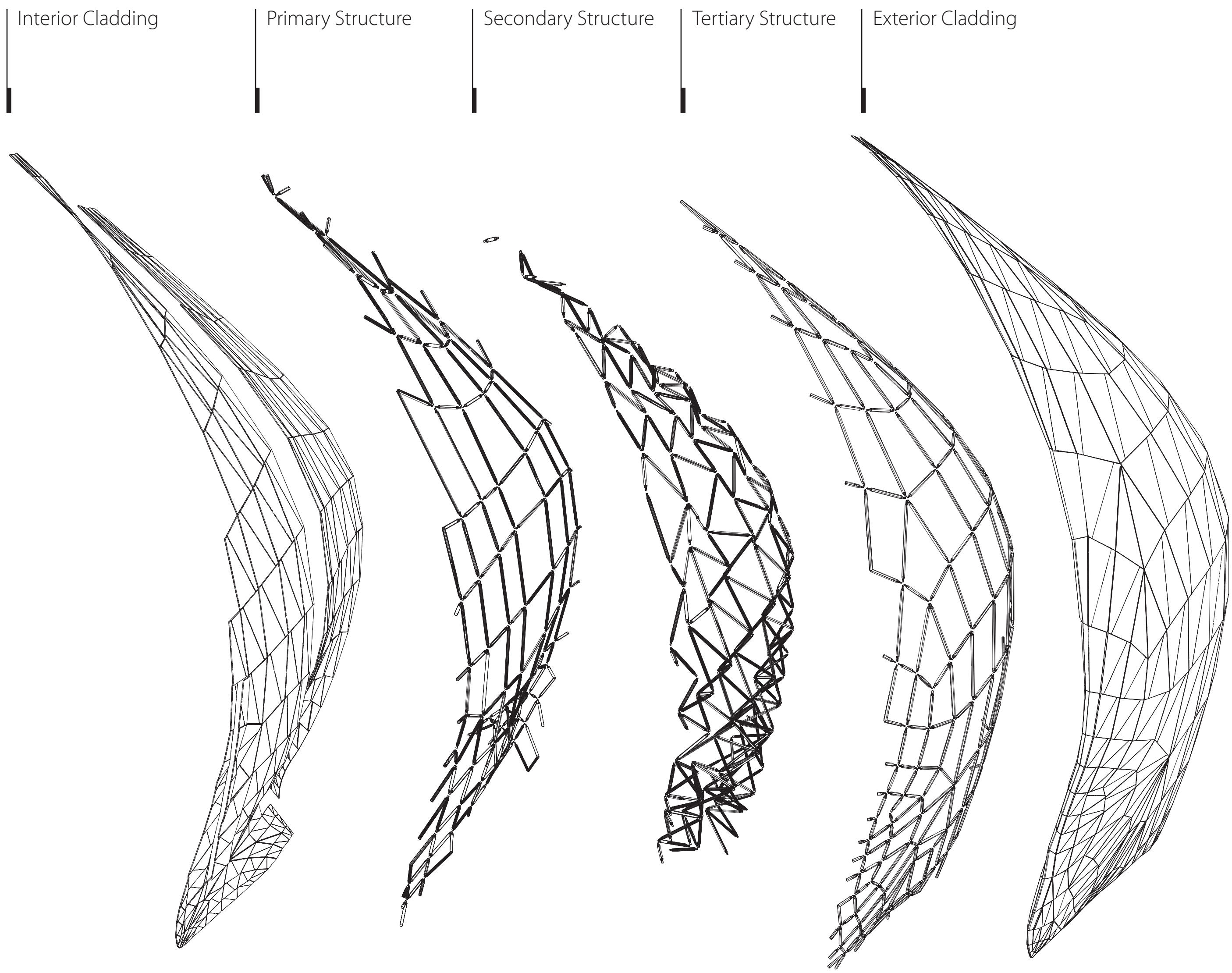
Structural Integrity

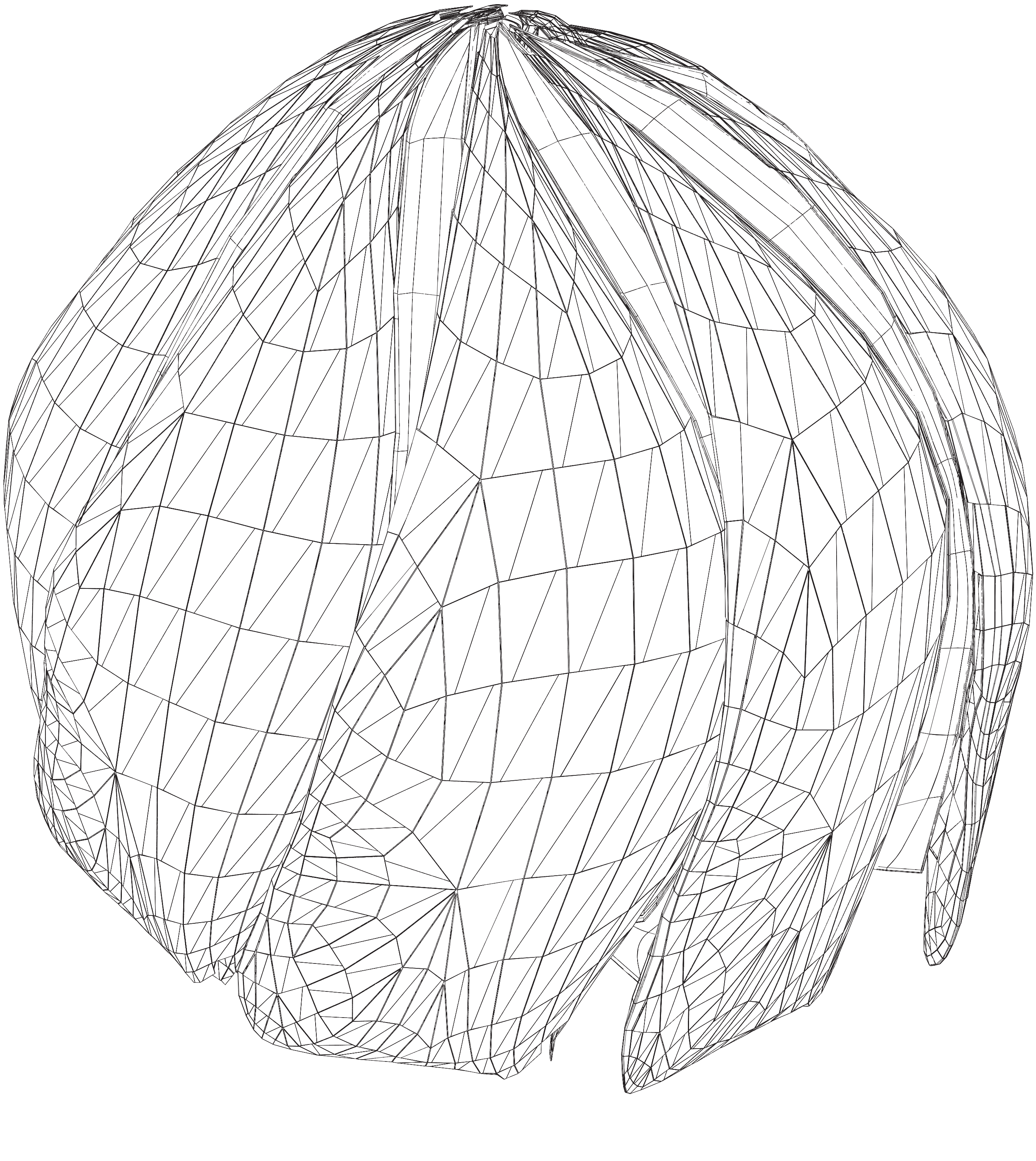
The fins are comprised of six layers of material and structure (see 2.05). The innermost layer is composed of large faceted triangles of alabaster stone. The alabaster is attached to the primary steel structure by steel framing, the steel framing functions similarly to a system of purlins and girders (seen in 2.11 and 2.14). The primary steel structure is composed of structural tubes that are connected by several large ball joints. The ball joints are poured from molten iron to form solid balls. The structural tubes are welded to custom cut iron plates, the plates are then welded to the ball joints. The secondary and tertiary steel structures are designed to inter-connect with the ball joints and support the large mass of the exterior cast-glass cladding. The cast-glass cladding is connected to the tertiary structure via the steel framing also used to secure the interior cladding. The physical characteristics of these building components is embedded within the Parts created in the CATIA model.

The space frame structure was designed to enhance the internal luminescence of the building as it is seen from the exterior. In reality the structure will be denser than originally anticipated due to the extreme dead load of the interior and exterior cladding. Wind loads play an important role in determining the structural integrity of the outer shell (the cast-glass cladding). The gaps between the panels of cladding are sealed with structural silicone. The services for the building are hidden within large aluminum tubes that run vertically throughout the steel space frame structure.

The facetization of the inner and outer cladding simplifies the manufacturing and design process, while reducing the overall costs of *machining* these materials. Using a large number of *complex curves* would increase the cost of fabrication and create excessive material waste. Both the cast-glass and the alabaster do not require any bending, only cutting and minimal *milling*.

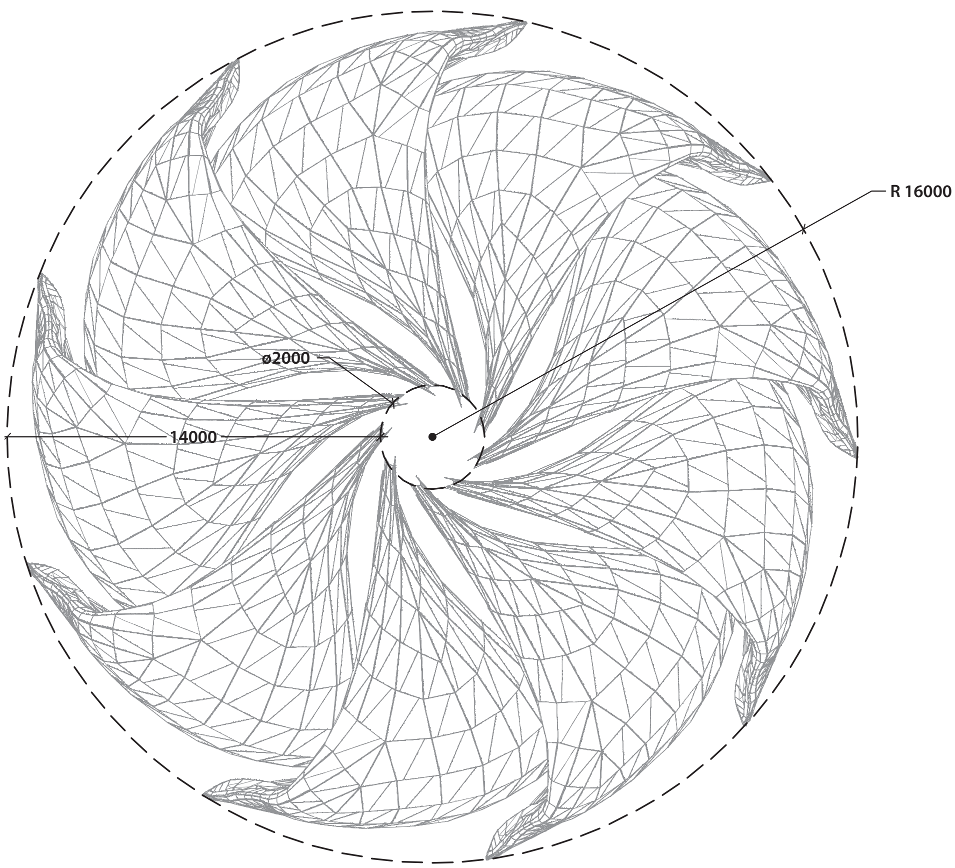
RIGHT *2.05 The following is an illustration of the layers of structure in the external Fins.*


Interior Cladding
Primary Structure
Secondary Structure
Tertiary Structure
Exterior Cladding



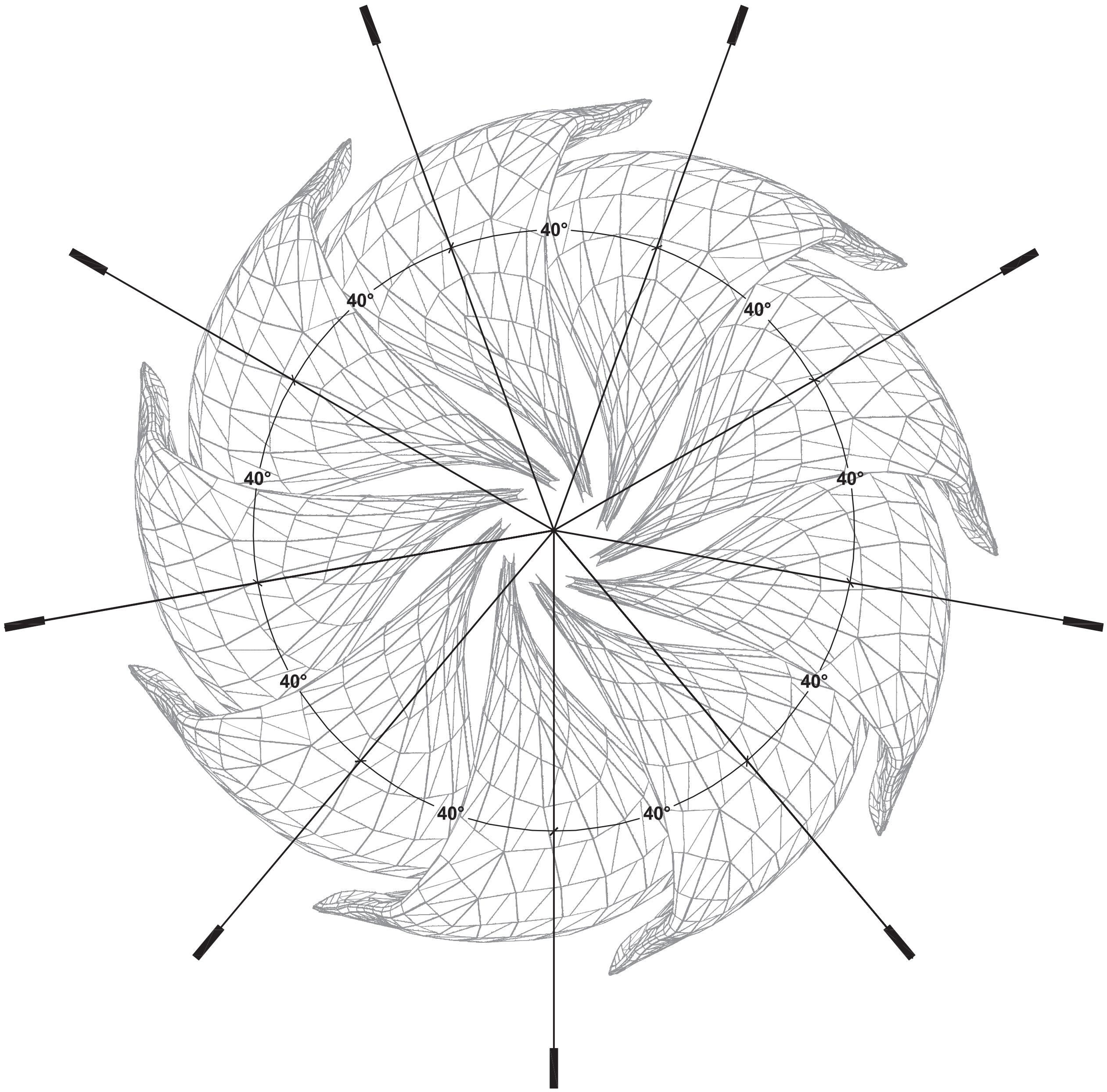
Axis of Rotation

The Temple fins are arrayed every forty degrees about a central axis. There are a total of nine entities, each comprised of the same components. The radius of the exterior at the base of the fins is approximately sixteen metres. The distance from the central opening at the peak of the fins to the extent of the structure of the skin is fourteen metres, and the diameter of the central opening is approximately two metres.

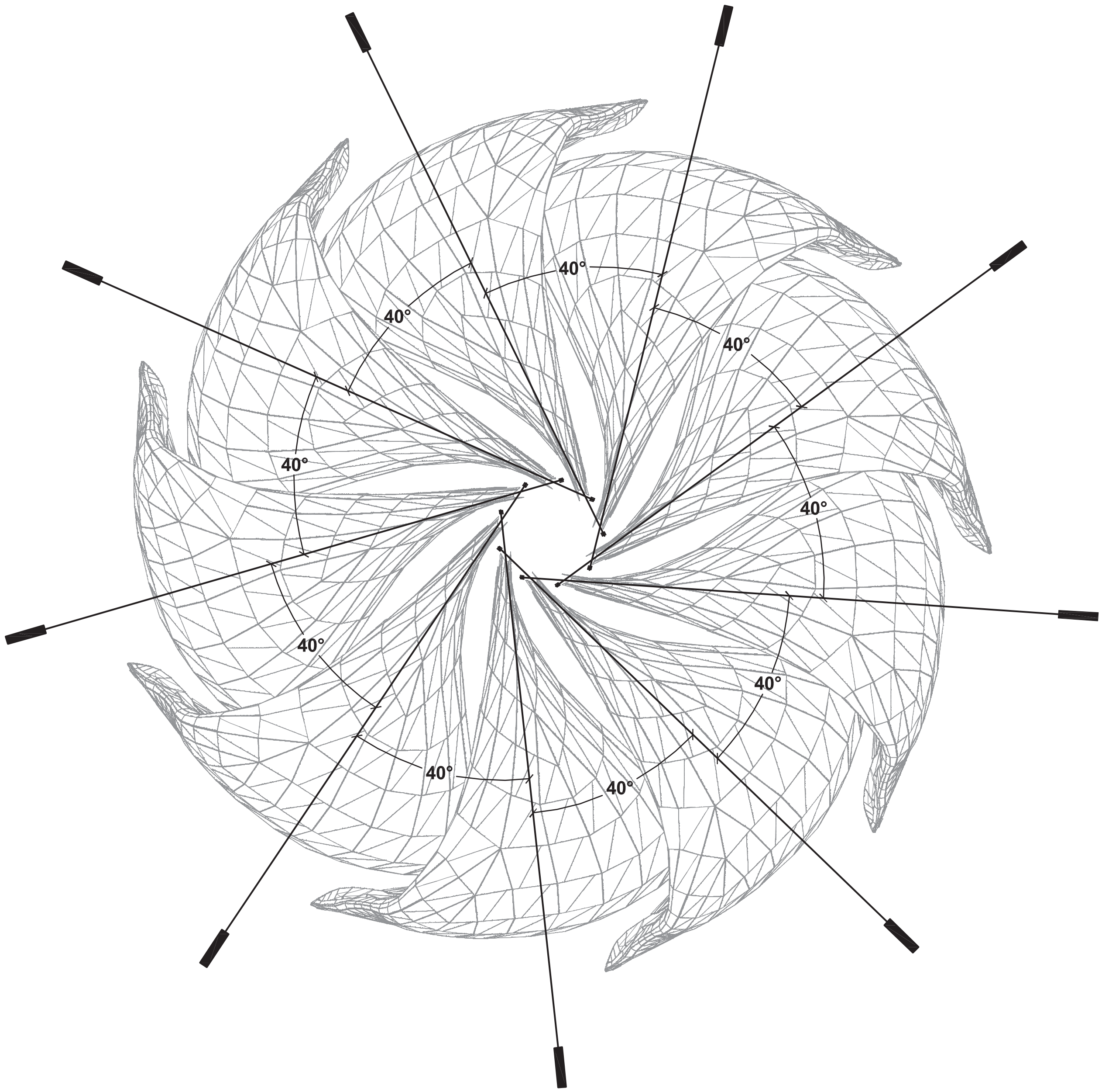


2.07 Crucial Temple dimensions

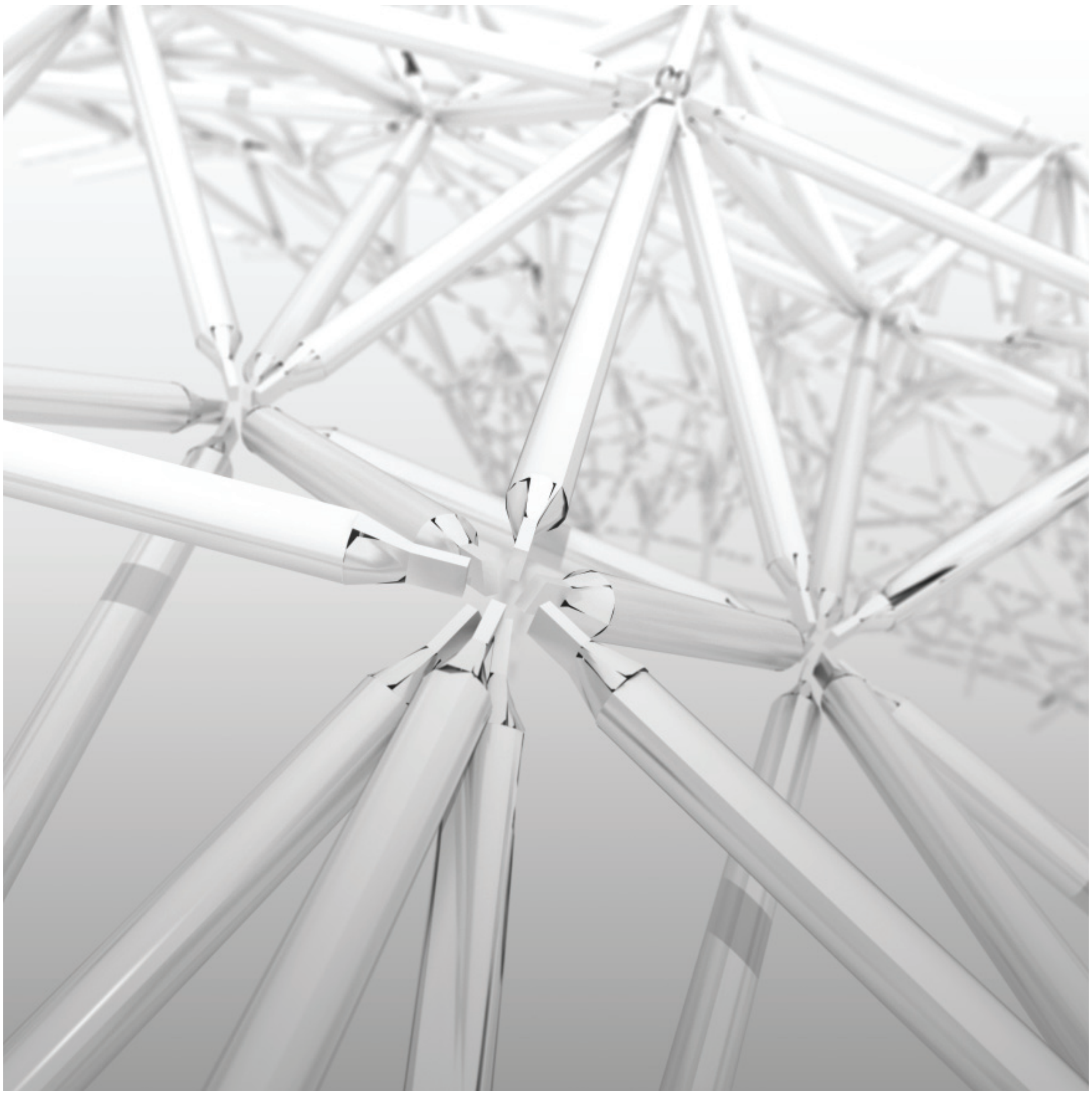
LEFT *2.06 The patterning fo the exterior glazed cladding system.*



2.08 The dimensions in this diagram are expressed in a radial pattern and revolve around the centre of the overall volume of the Temple



2.09 The dimensions in this diagram are expressed in a radial pattern and are drawn along the centre lines of each of the nine Fins



2.10 Structural space frame detail of a Temple Fin

Part Detailing

Several components of the Bahá'í Temple for South America contain a complex narrative. The exploded structural fin diagram (see 2.17) describes one example of a series of complex layered components. The structural system of the fins is broken down into several components, which are further separated into a series of sub-components.

The primary components of each fin (as described in 2.05) are:

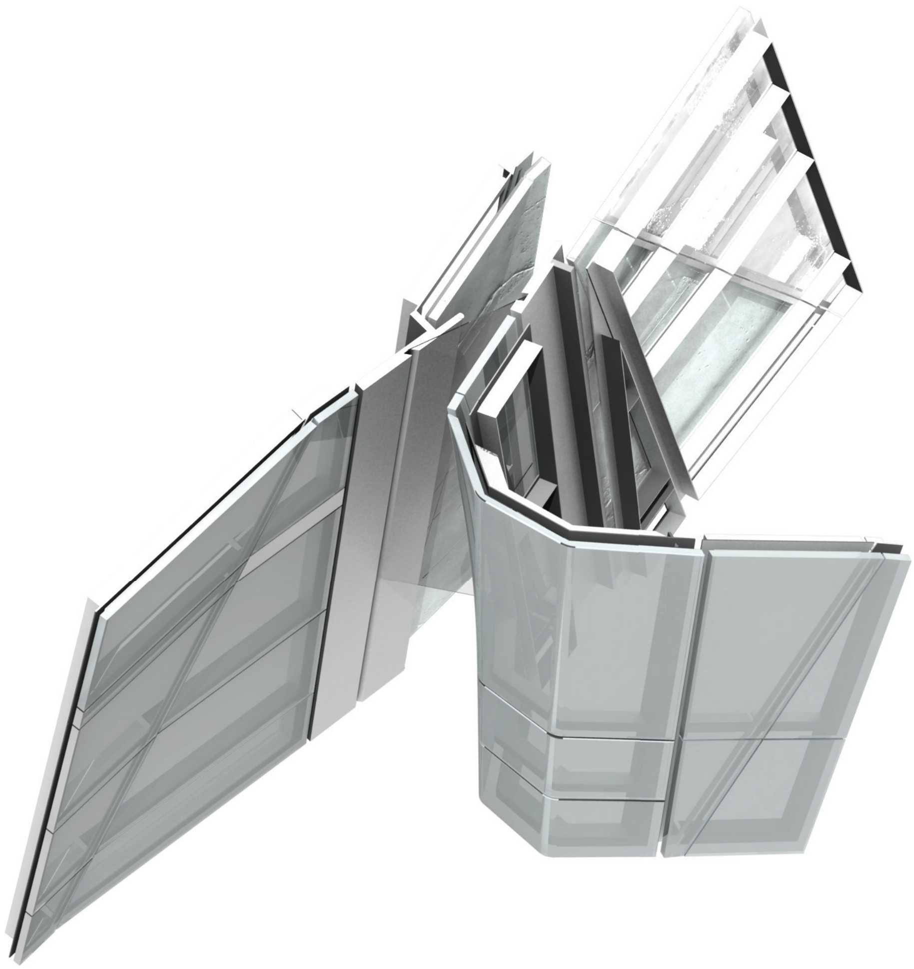
- Interior Cladding (Sealed Alabaster Stone)
- Space Frame Structure (Structural Steel Tubes)
- Exterior Cladding (Cast-Glass)

The primary components (seen in 2.05 and 2.17) consist of the following sub-components:

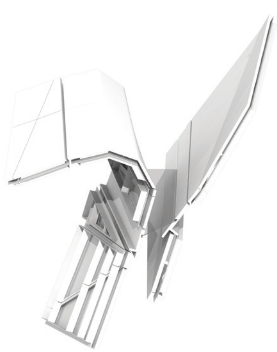
- Steel Framing
- Primary Steel Structure
- Secondary Steel Structure
- Tertiary Steel Structure
- Ball Joints
- Iron Plates

The application of these components is unique, the components are not. The use of steel framing employs familiar construction methodologies. Since the majority of the surfaces are divided and triangulated the structure is somewhat simple to determine. With little modification the *grid lines* are pre-existent. Part of this is due to the intrinsic attributes of facetization as well as the pattern; the pattern is a crucial structural component. The shape and size of the facets are a direct correlation of the Temple's structural requirements. Each section of steel framing is cut to fit (seen in structural details 2.11-2.16), nine duplicates are made of each cut to complete the remaining Fins.

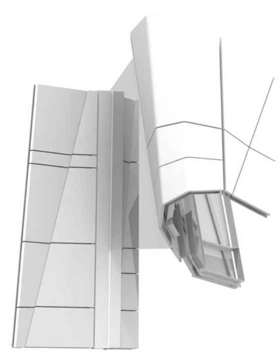
Once the basic form is established (sculpturally, methodically or otherwise calculated), a part schedule is generated to determine the placement of the building's many components.



2.11 Structural detail expressing materiality where Fins overlap (Front Perspective)

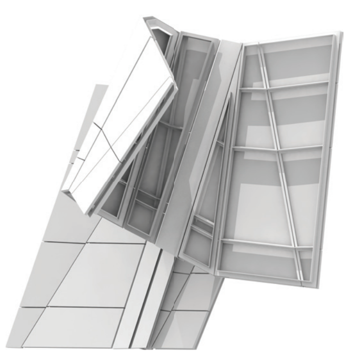


2.12 Structural Detail (Top)

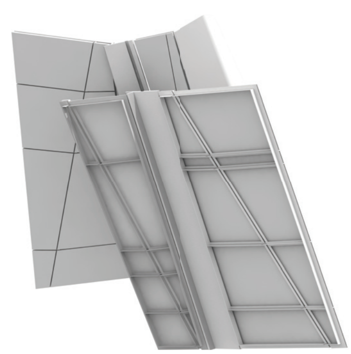


2.13 Structural Detail (Front)

2.14 Structural detail expressing materiality where Fins overlap (Reverse Perspective)



2.15 Structural Detail (Side)



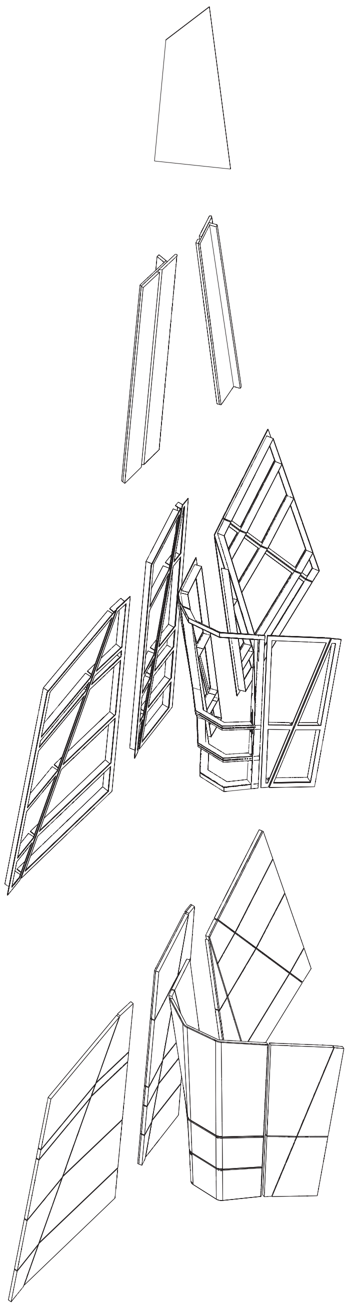
2.16 Structural Detail (Side)

Glazing Panels
Irregularly-shaped glazing panels spanned between overlapping areas of arrayed fins. Clear glass, used to dampen the force of circulating air-flow and wind loads.

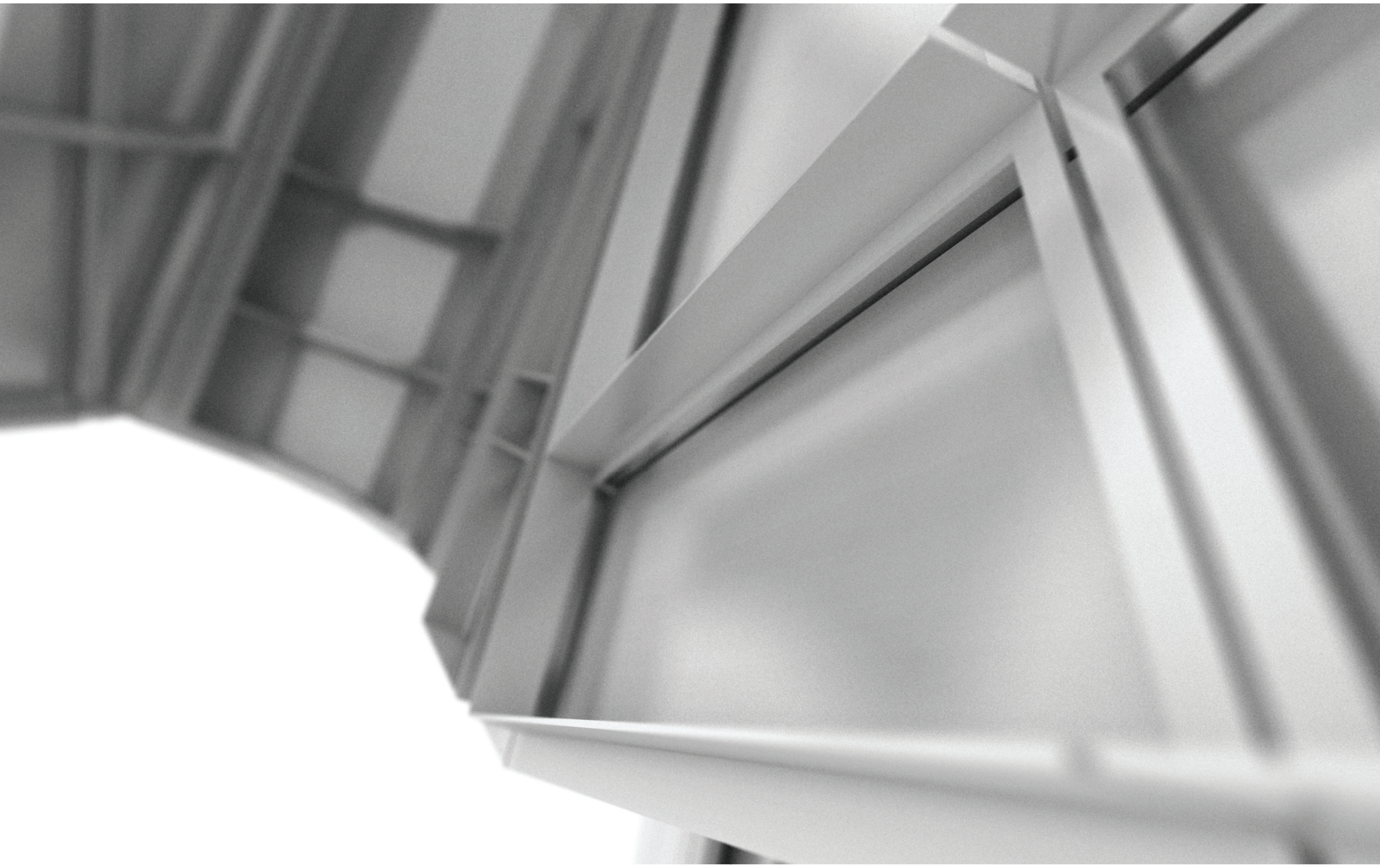
Bronze Cast Mullions
The mullions are cast in several sections and correspond to the glazing panel schedule. They are connected to the steel framing by rectilinear steel flanges.

Steel Framing
Steel framing supports the mullions and gives the primary, secondary and tertiary structure something to adhere to. It supports both interior and exterior cladding systems. The steel framing is fragmented by partitions that correspond to the schedule of both types of cladding.

Interior/Exterior Cladding
The interior cladding is comprised of large fragmented components of alabaster stone, while the exterior cladding system consists of several pieces of cast-glazing. Both systems adhere to the system of structural steel banding beneath.



2.17 Exploded detail revealing cladding and mullion construction conditions where Fins overlap



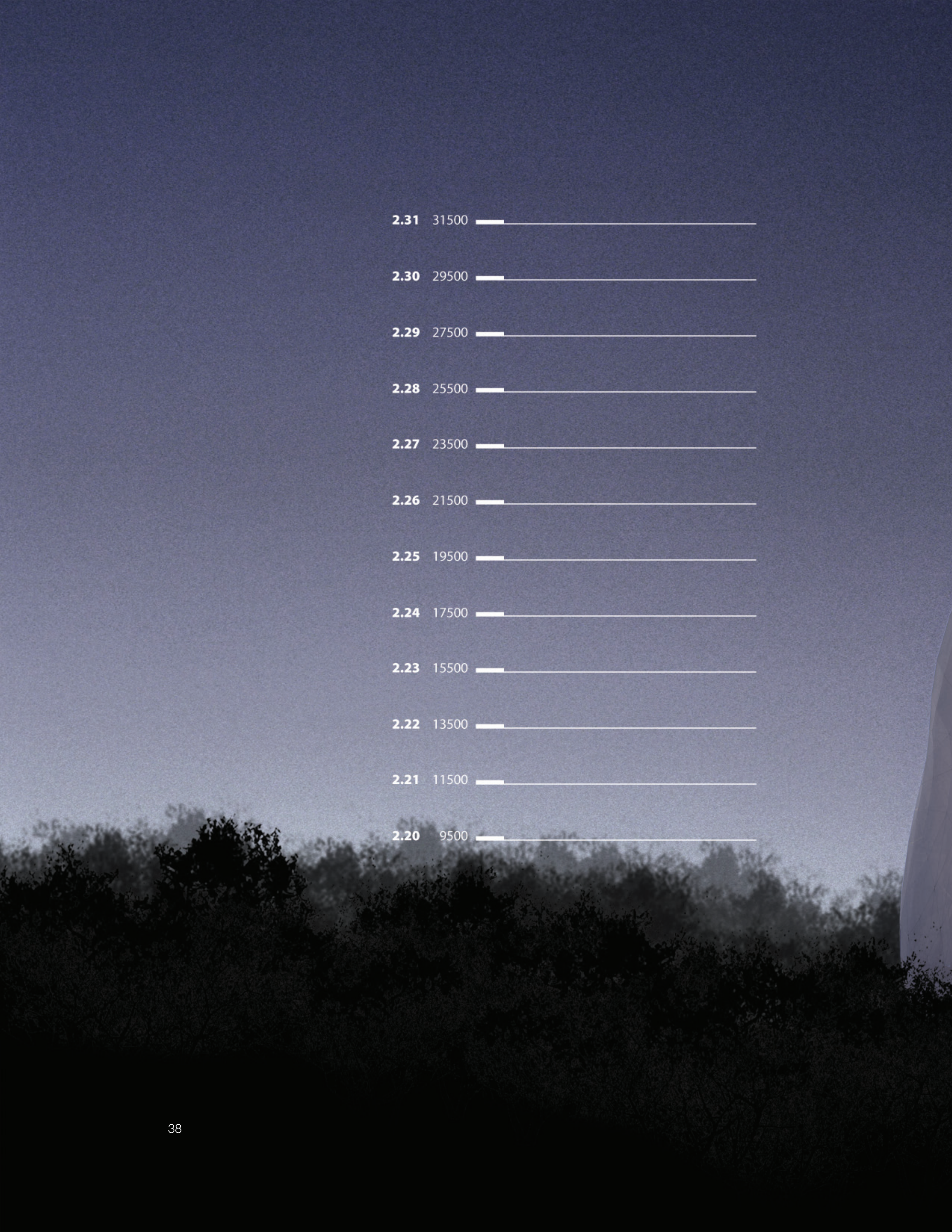
2.18 Detail that illustrates how the structural steel banding is attached to the interior cladding

SKIN & STRUCTURE

Sectional Detailing

The following static figures provide a sense of animation, and describe a vertical shift in complexity. The sections are cut horizontally in an orthogonal manner, revealing the luminescent qualities of the exterior skin which is meant to "dissolve [and] reappear in light."[4]

NEXT *2.19 Conceptual rendering of the Temple exterior in context.*


2.31 31500
2.30 29500
2.29 27500
2.28 25500
2.27 23500
2.26 21500
2.25 19500
2.24 17500
2.23 15500
2.22 13500
2.21 11500
2.20 9500

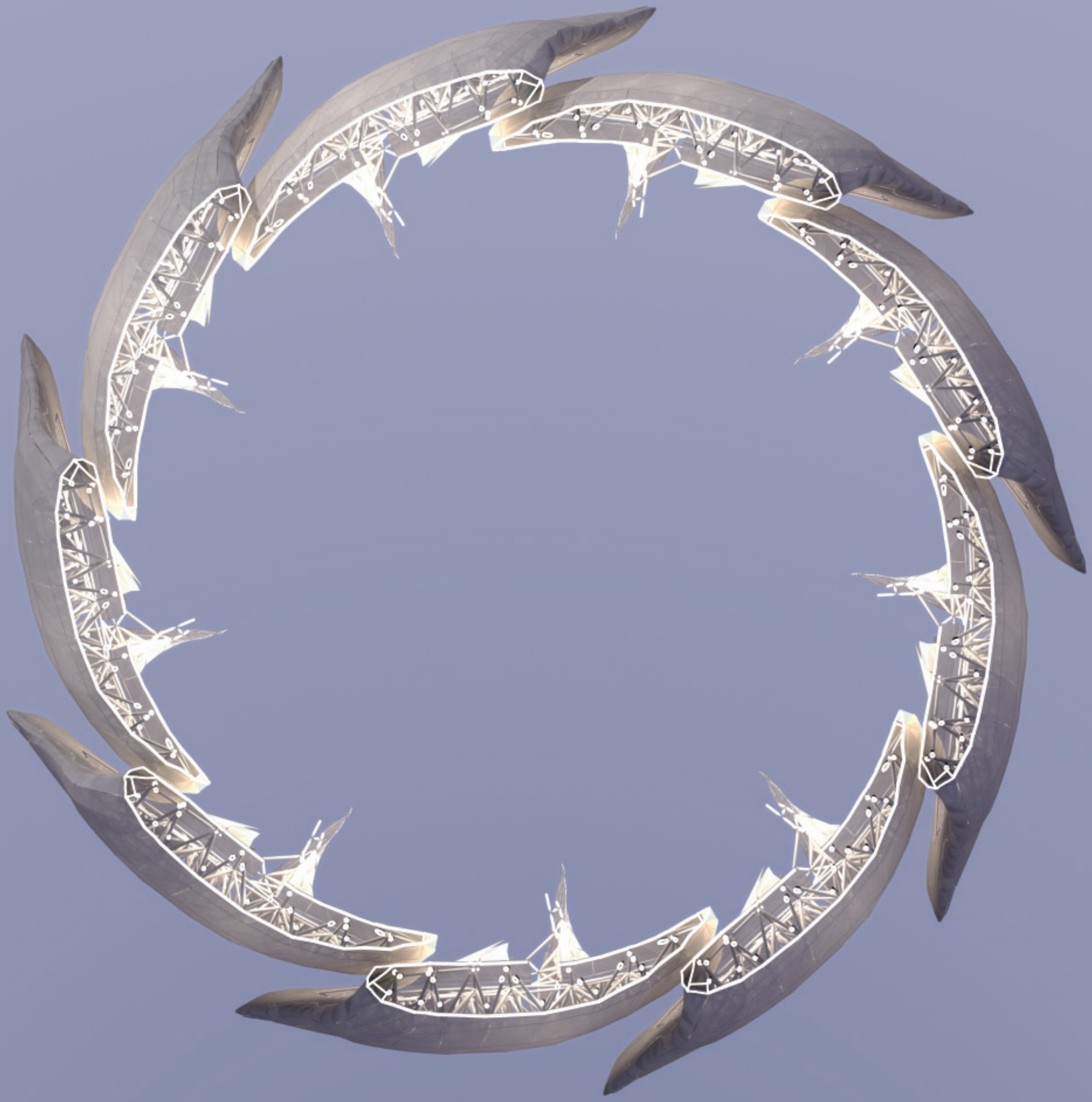
2.20 Planametric Section at 9500mm Above Grade

2.21 Planametric Section at 11500mm Above Grade

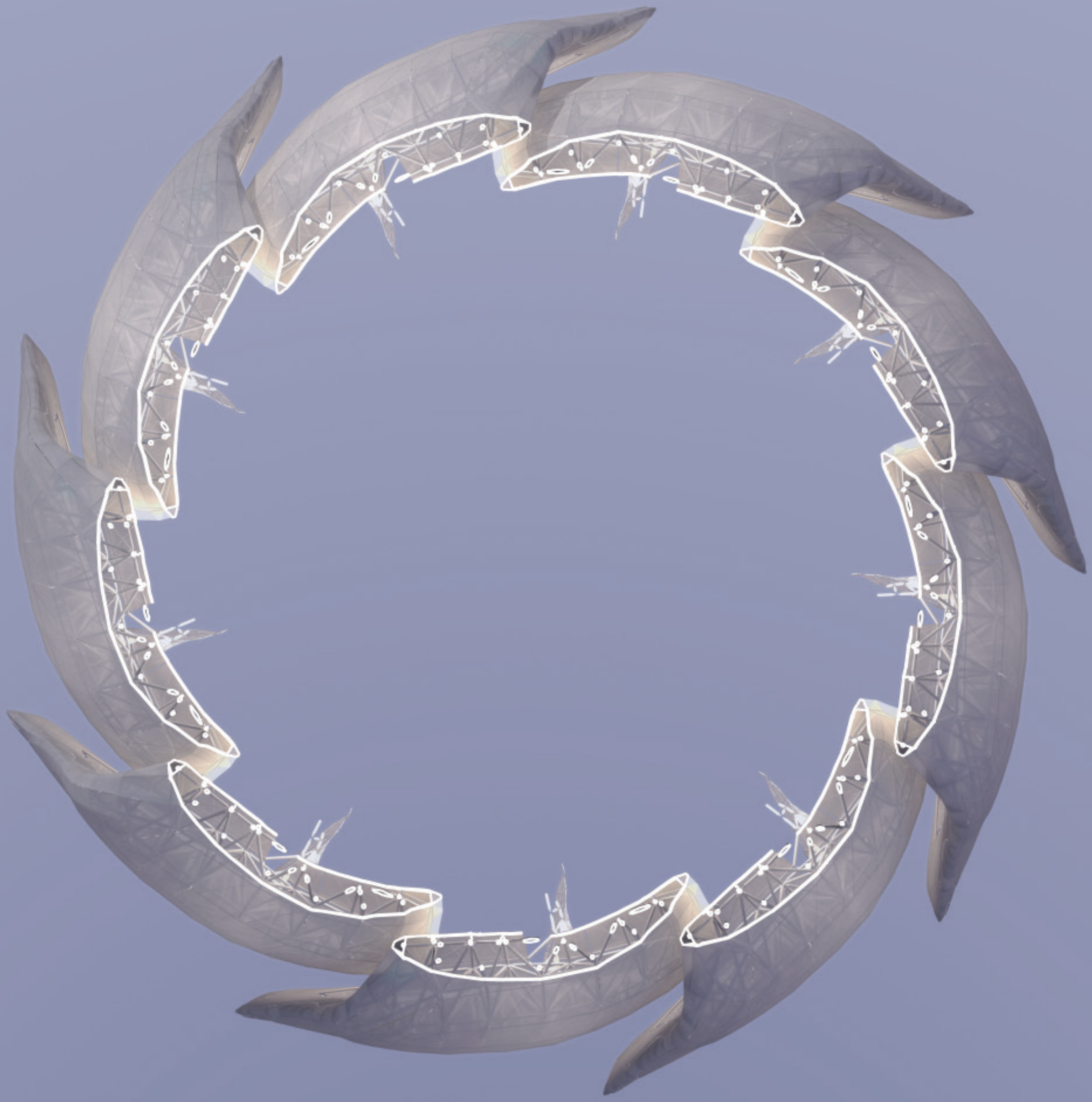
2.22 Planametric Section at 13500mm Above Grade

2.23 Planametric Section at 15500mm Above Grade

2.24 Planametric Section at 17500mm Above Grade

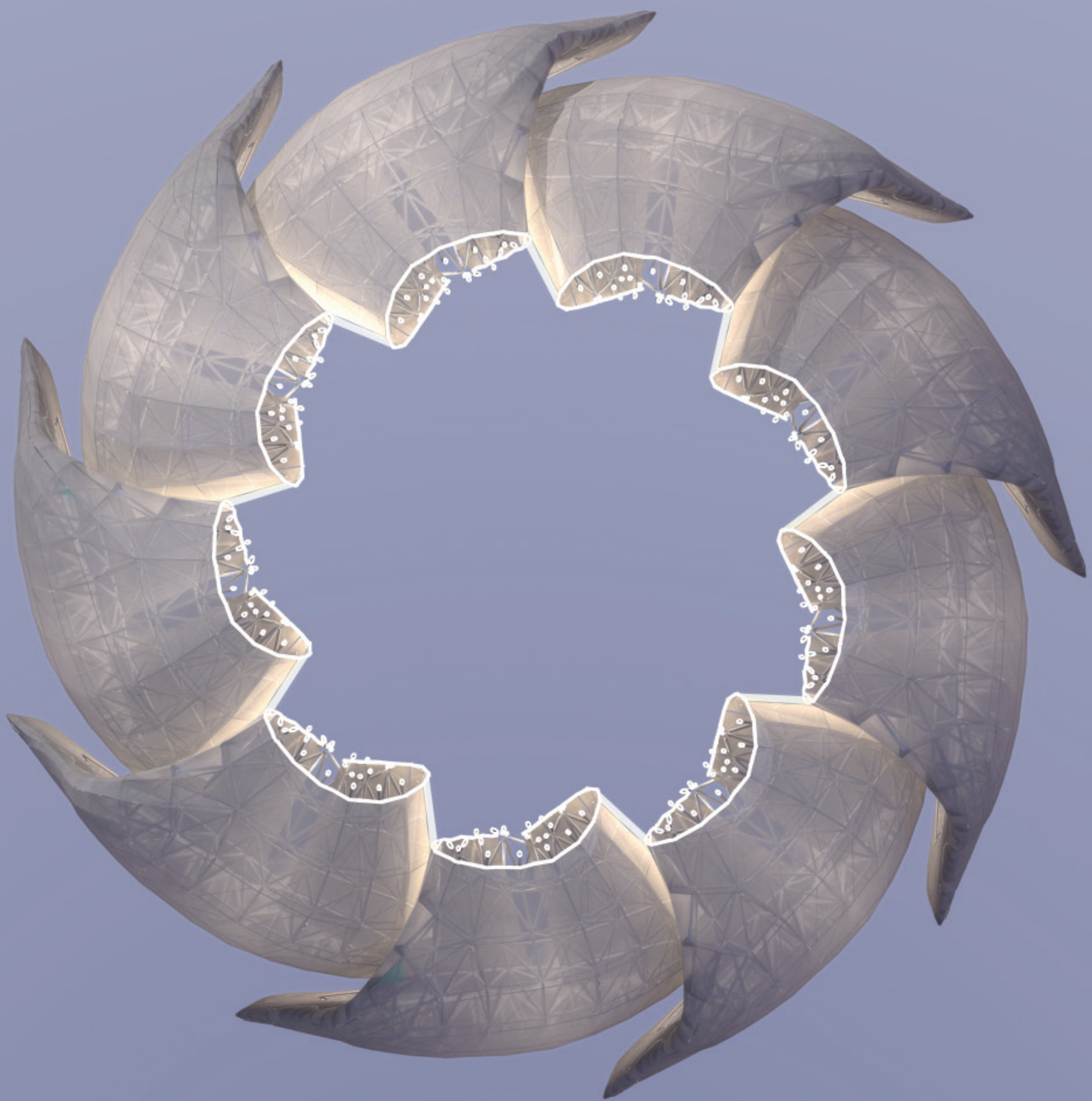


2.25 Planametric Section at 19500mm Above Grade

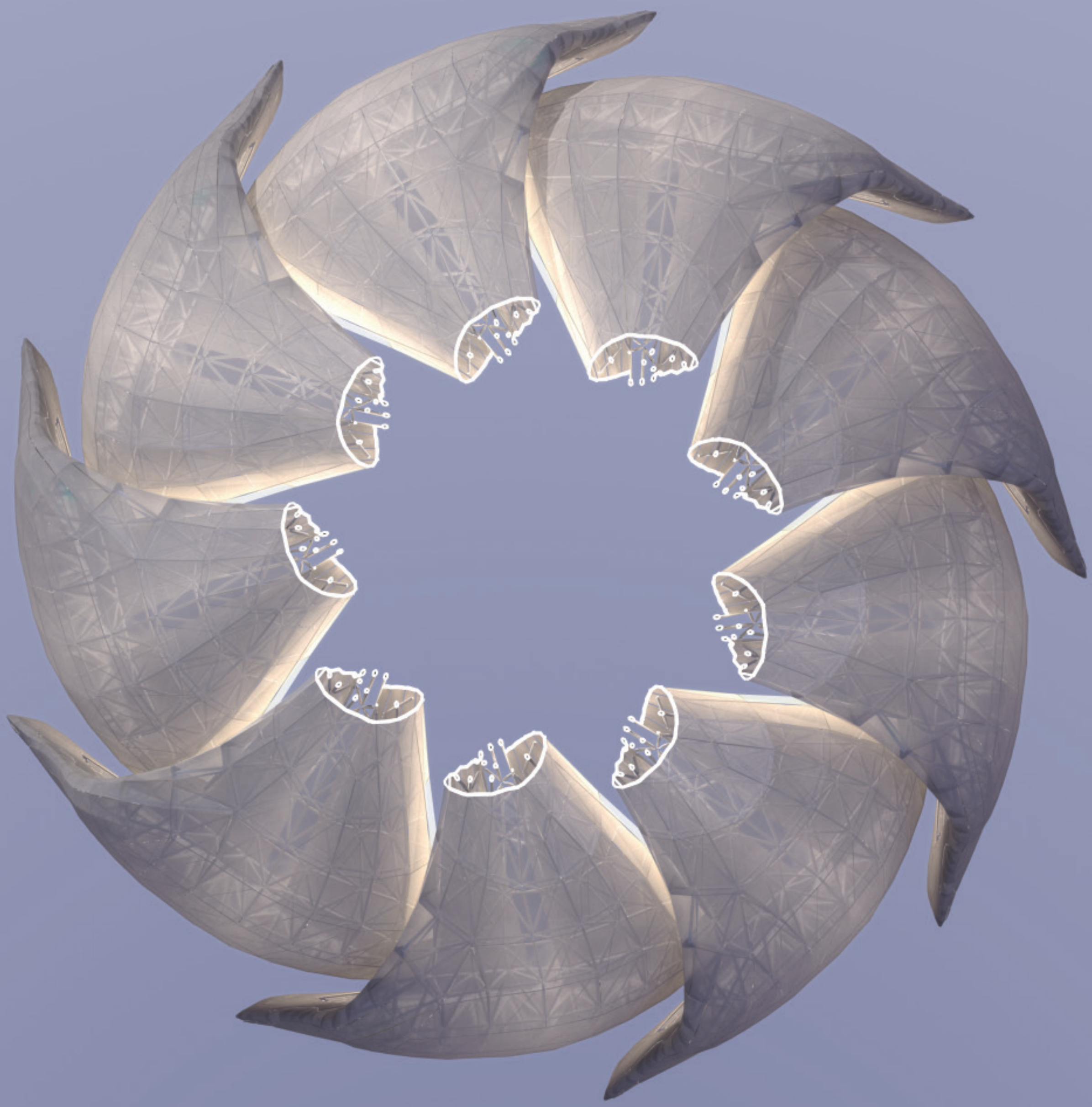


2.26 Planametric Section at 21500mm Above Grade

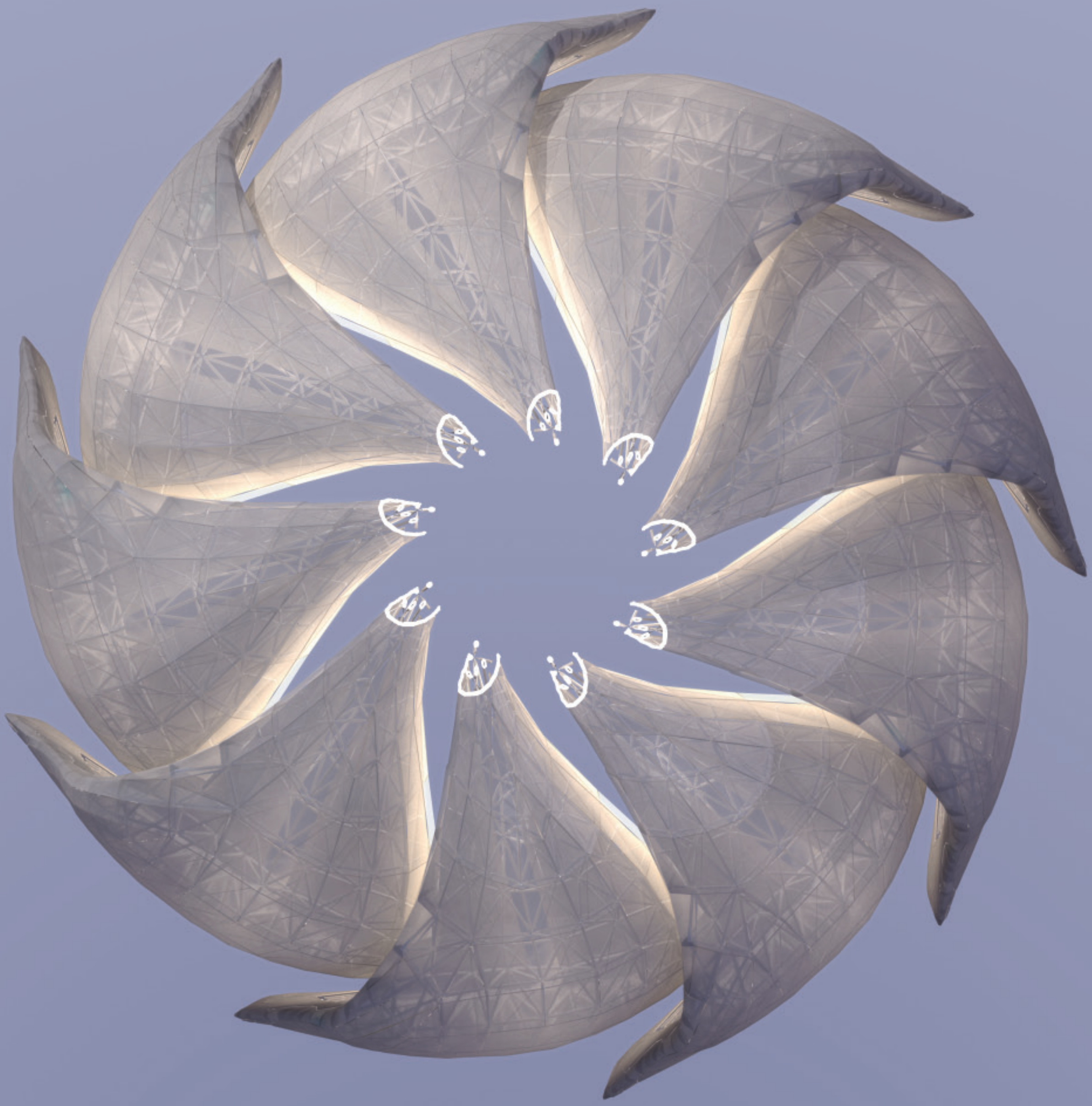
2.27 Planametric Section at 23500mm Above Grade



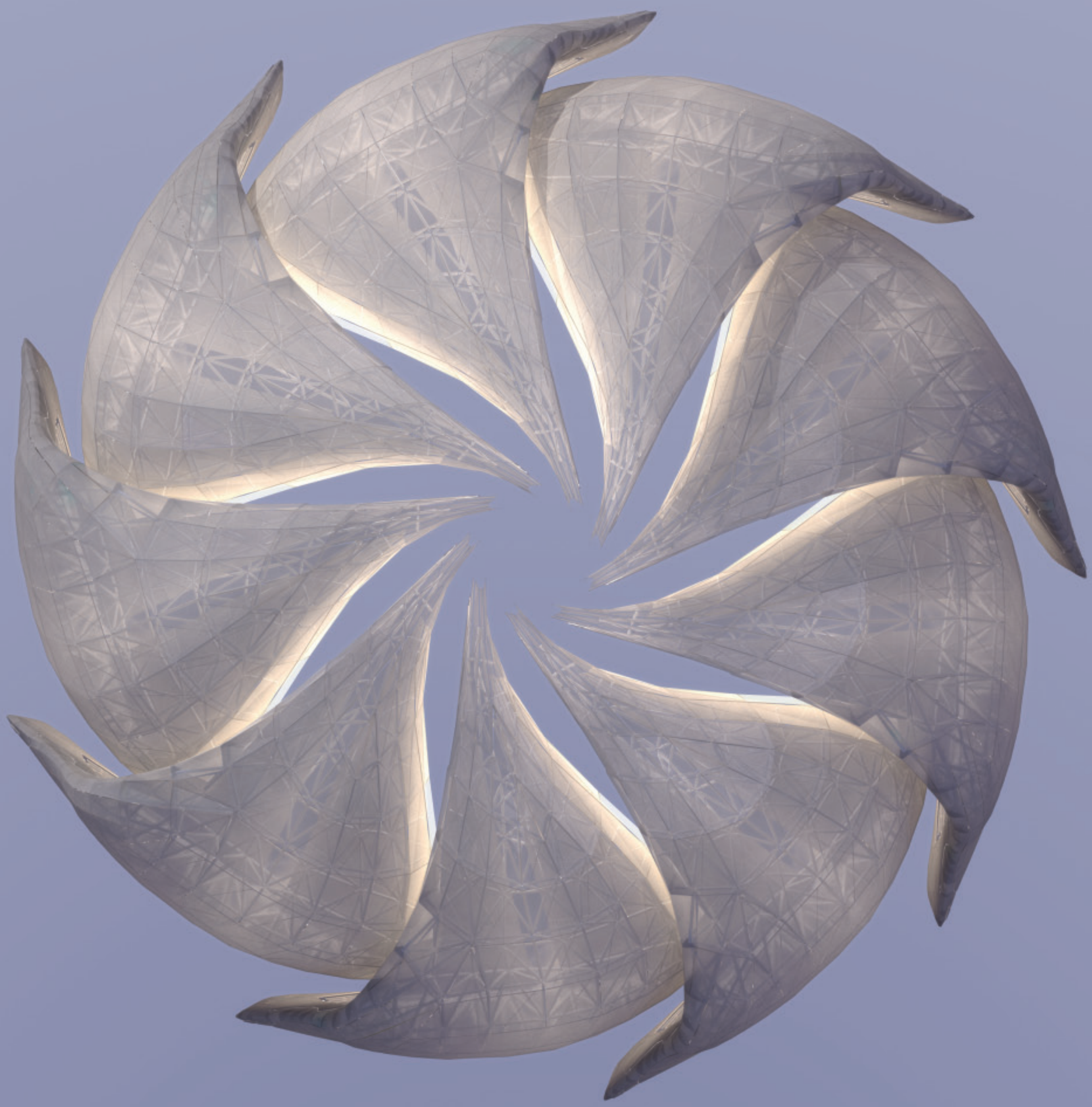
2.28 Planametric Section at 25500mm Above Grade



2.29 Planametric Section at 27500mm Above Grade



2.30 Planametric Section at 29500mm Above Grade



2.31 Planametric Section at 31500mm Above Grade

PROGRAMMABLE ANALYSIS

Sorting Digital Data

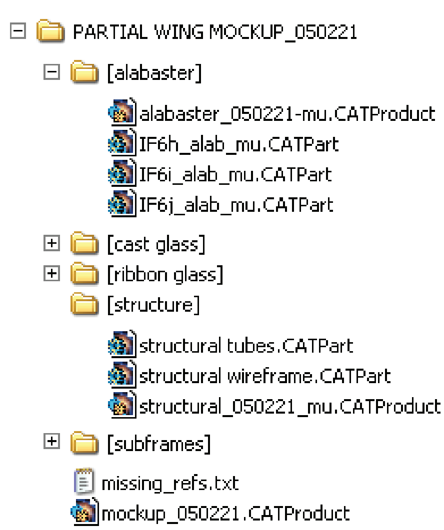
Software advances have drastically changed the way we design and build architecture.[5] The more information we are able to procure, the more documented buildings become. There is an overwhelmingly large responsibility of *information management* inherent in the development of this data. While generating adequate documentation to describe the assembly of a single product or component has become routine; the challenge as it stands, is how to organize data when you have hundreds of thousands, or even millions, of parts to assemble.

This process demands a level of self-discipline on the part of the designer. Naming conventions, object grouping, layer organization, and general file maintenance, become integral components of the design process. Information is useless without context.

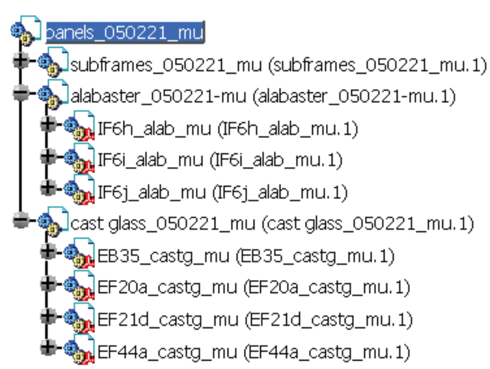
Many systems allow for the *bidirectional* exchange of data, however few of them allow data to be readily accessible. Most platforms offer access to information through commands. In some extreme cases the only way to generate certain types of data is through *scripting* or *programming* (a very systematic approach), while typically reserving the *visual interface* for common, routine tasks. For the platform to be effective, systematic data needs to be visually accessible. The easier it is to generate and extract data, the simpler it is to sort and maintain the flow of information.

CATIA is a powerful modeling tool, it can administer very large *data-sets*. In CATIA, the structure of the three-dimensional model is what determines file layout and location (see 2.32 and 2.33). It uses a hierarchical approach to structural organization. It groups models under two major headings, Parts and Products. A Product can consist of several Parts or other Products. CATIA is a *Product Lifecycle Management (PLM)* solution commonly used in aerospace engineering, industrial product manufacturing, ship building and vehicle manufacturing; it is a *robust* and complex tool that has a very strong data management utility known as the Desk. The Desk allows you to edit data structures such as file location and Part/Product *associativity*.

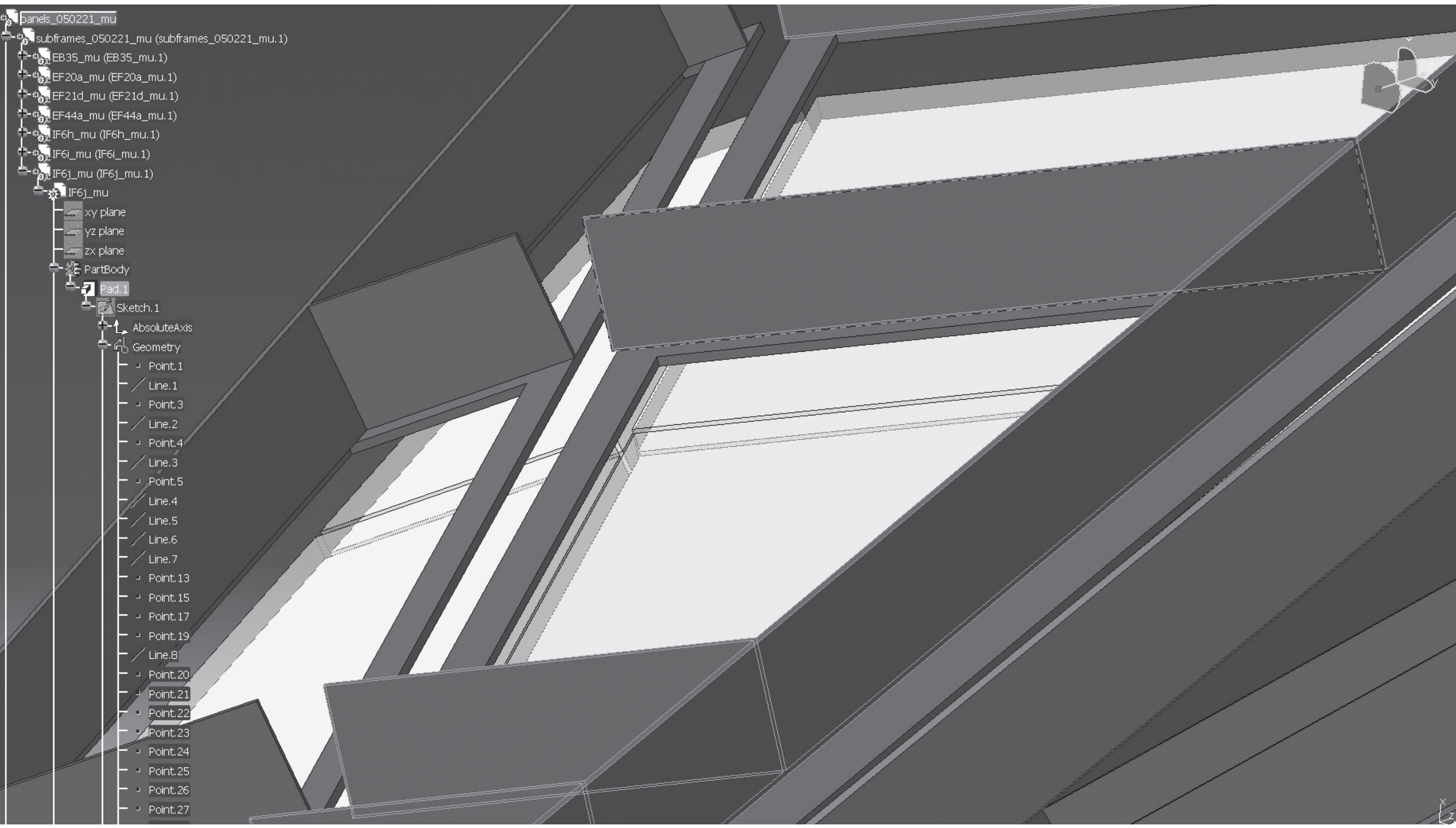
Visually organizing data allows us to comprehend the construction of a model, just as drawings allow us to visualize its underlying composition. The following illustrations (2.32 and 2.33) compare the interface of Windows Explorer—something common to almost every user—to the layout of the *Specification Tree* within CATIA.[6] Careful consideration has been taken to transcend a conventional workflow and make organizational aspects of the product familiar to the user. The breadth of the content stored within the Specification Tree is what makes CATIA such a powerful tool, every single aspect of an object's construction, linkage or materiality is stored within the tree. Tree objects can be modified and reassigned at any point throughout the modeling process.



2.32 The Windows Explorer browser allows you to locate and organize your files



2.33 The CATIA Specification Tree distributes your modeling components into collapsible branches, the root branch is the product, all sub-branches contain CATIA parts—further sub-branches store part construction and physical properties



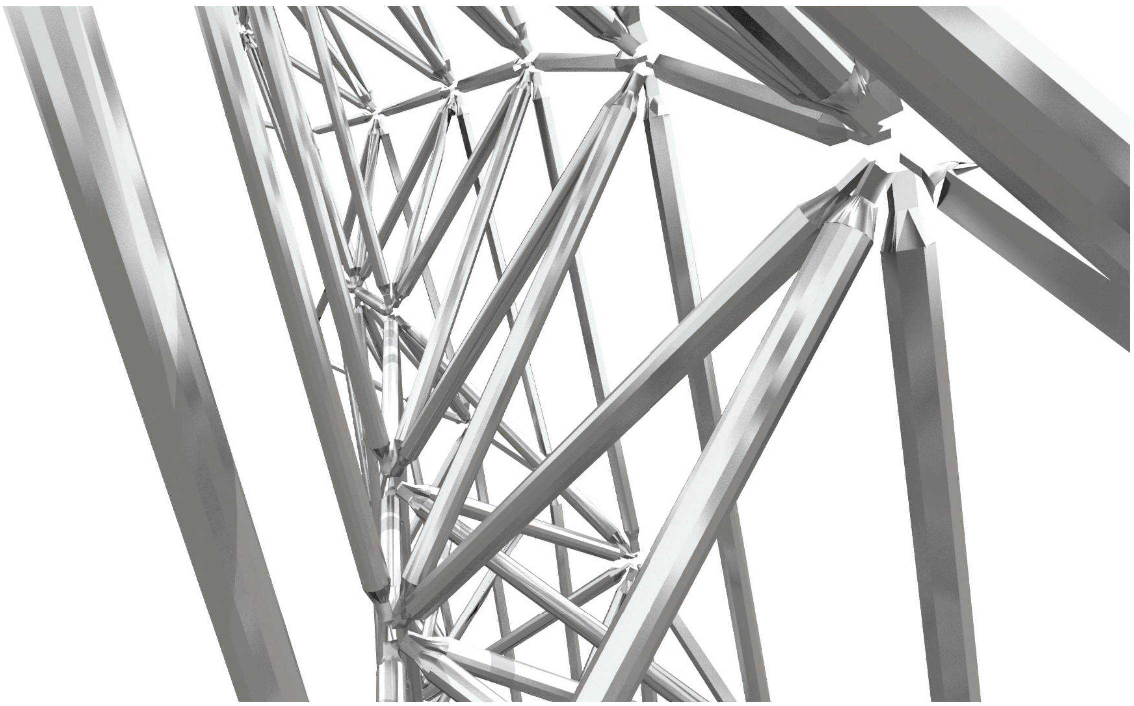
2.34 CATIA's hierarchical tree stores all properties associated with a specific part or component

Digital Detailing

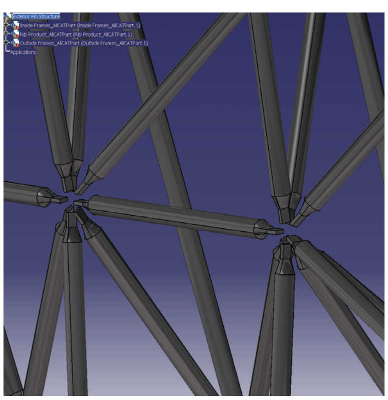
The following figures are graphical expressions demonstrating the effects of the exterior and interior alabaster cladding on the underlying space-frame structure. The deflection is minimal (0.0123 inches). The space frame structure distributes the stress in an effective pattern, almost entirely even from member to member. Since the members are allowed to deflect (to a certain degree), they cannot be considered rigid members and thus transfer wind resonance effectively.

CATIA allows the user to interactively perform variable stress tests by selecting members by material. When you assign a material to an object or model you assign all its physical attributes, as opposed to visualization software which assigns aesthetic properties. The tests that were performed in figures 2.35 through 2.43 were performed with the chemical composition of standard steel members, the diagrams depict the results of the dead load of the structure. Since the connection of these members plays an exceptionally large role in how they manage deflection, two entirely unique joints were used in the test. The Plated joint (see 2.44) restricts lateral movement and torsion and transfers forces to stress resistant plates through rigid rubber dampers. The rigidity of the joint caused each connecting member to fail at its corresponding connection, shearing the member at several points. The initial size and mass of the Plated joints were grossly underestimated. A preliminary plate thickness of 15mm required reconsideration and was eventually increased to 30mm, increasing the overall mass of the joint by 21.1952 kg.[6] The performance of the Spherical joint (see 2.45) far exceed that of the Plated joint.. Although it nearly failed the test (and may fail when considering wind loading), its overall performance exceeded that of the first joint. The Spherical joint weighed in at a graceful 6.1655 kg, an important figure considering there are well over 2,600 joints throughout the space frame structure. Using this joint reduces the overall net weight of the joints by approximately 30,080 kg. Manufacturing the joint with titanium further reduces the net weight by 6,800 kg and brings the weight of a single joint down to 3.5451 kg.

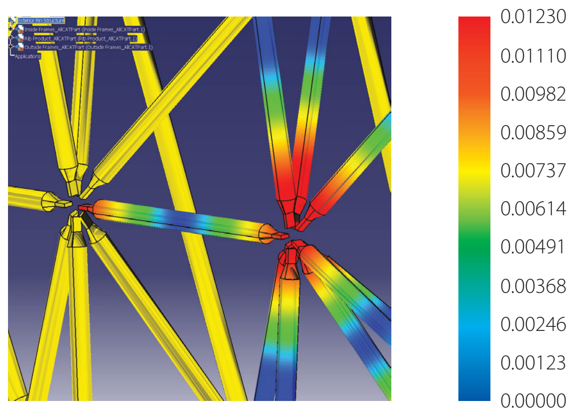
These joints are an educated projection of what form a final joint might take. Judging by the stress test of the second joint, further development could yield a result that would satisfy more precise stress conditions and generate an optimal connection.



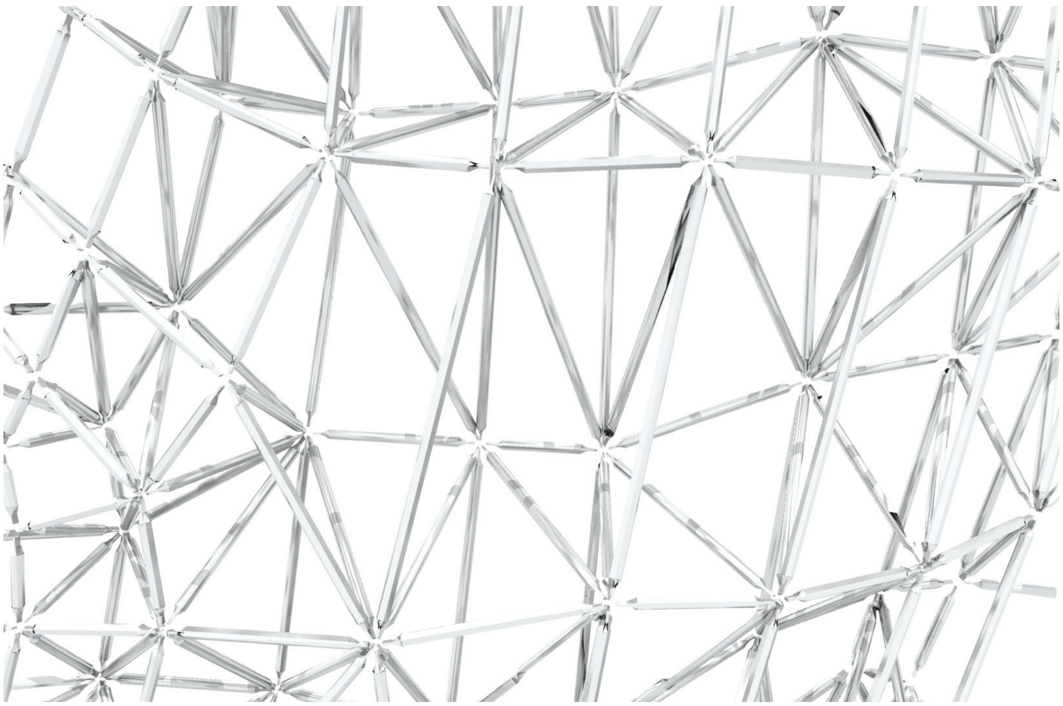
2.35 Structural Truss Rendering



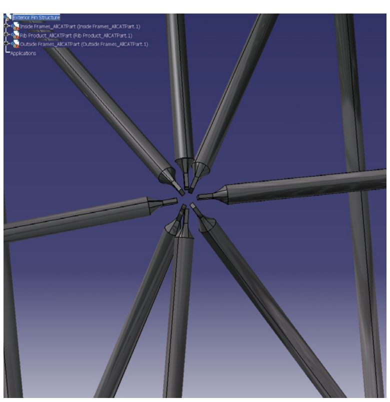
2.36 Truss Models, CATIA



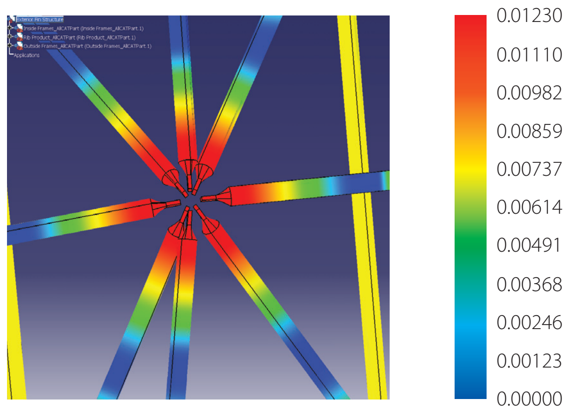
2.37 Deflection Diagram, CATIA (in.)



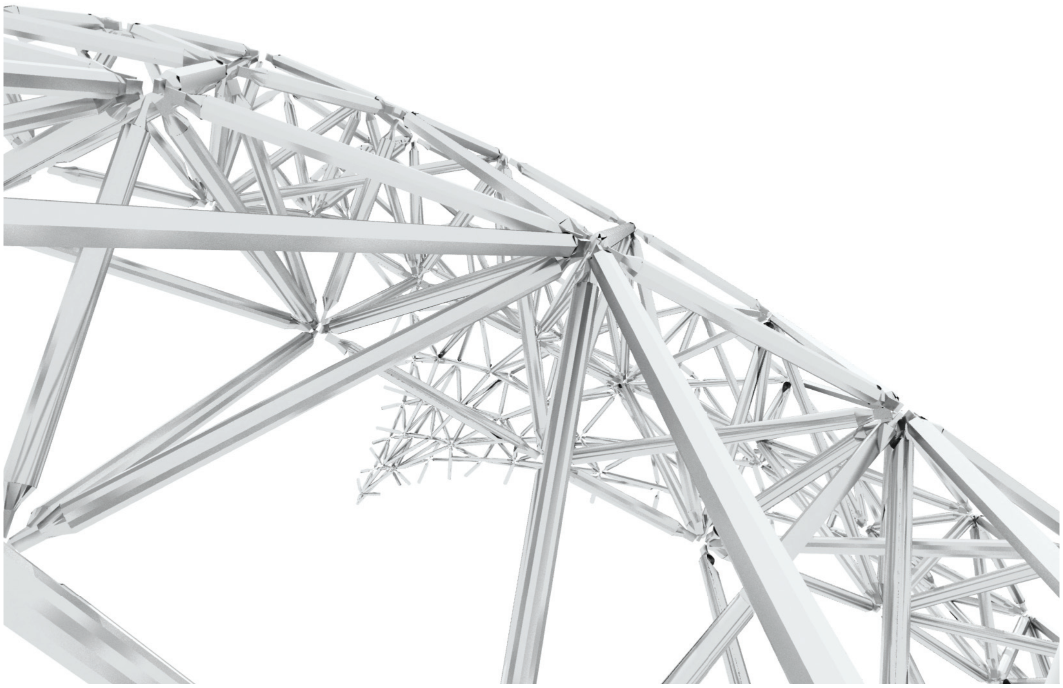
2.38 Structural Truss Rendering



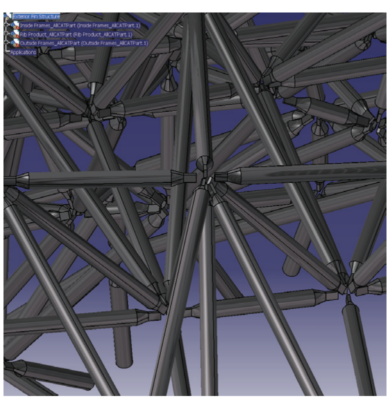
2.39 Truss Models, CATIA



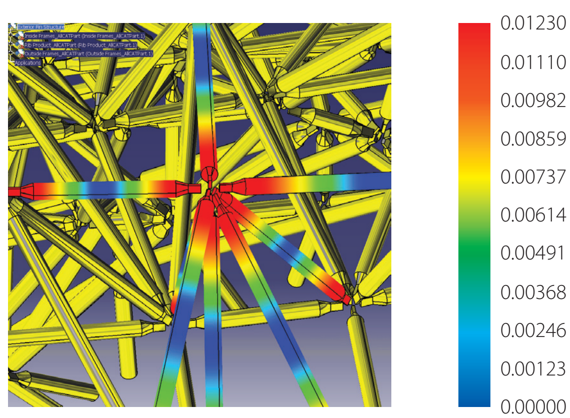
2.40 Deflection Diagram, CATIA (in.)



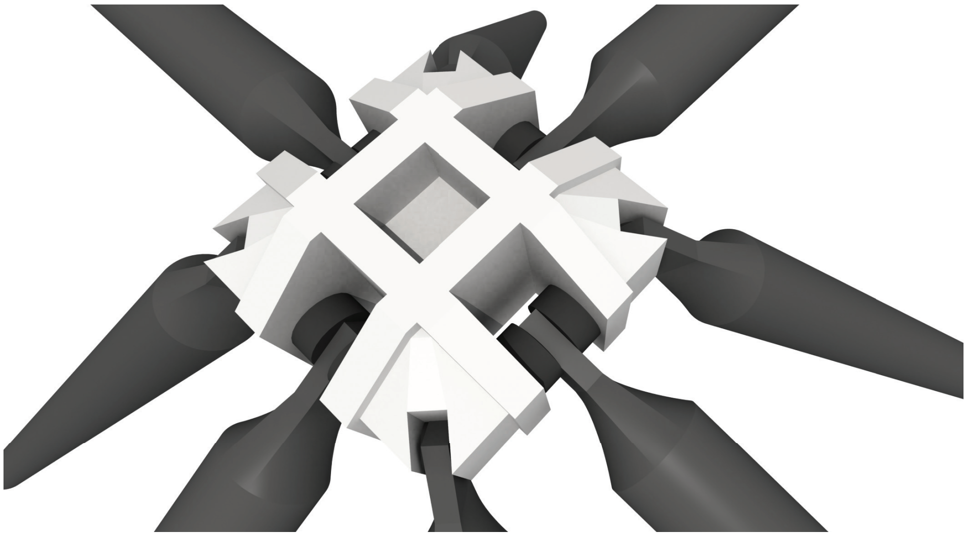
2.41 Structural Truss Rendering

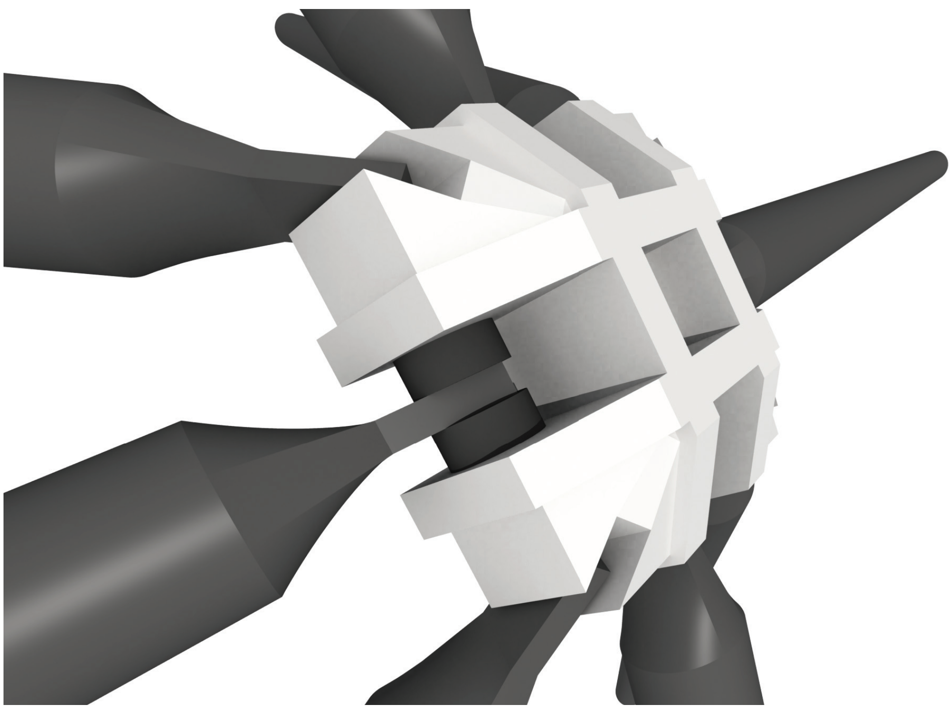


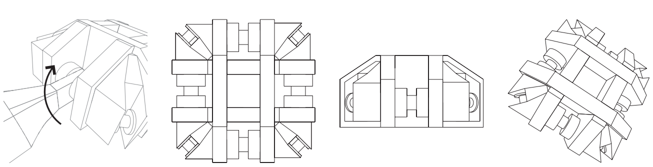
2.42 Truss Models, CATIA



2.43 Deflection Diagram, CATIA (in.)

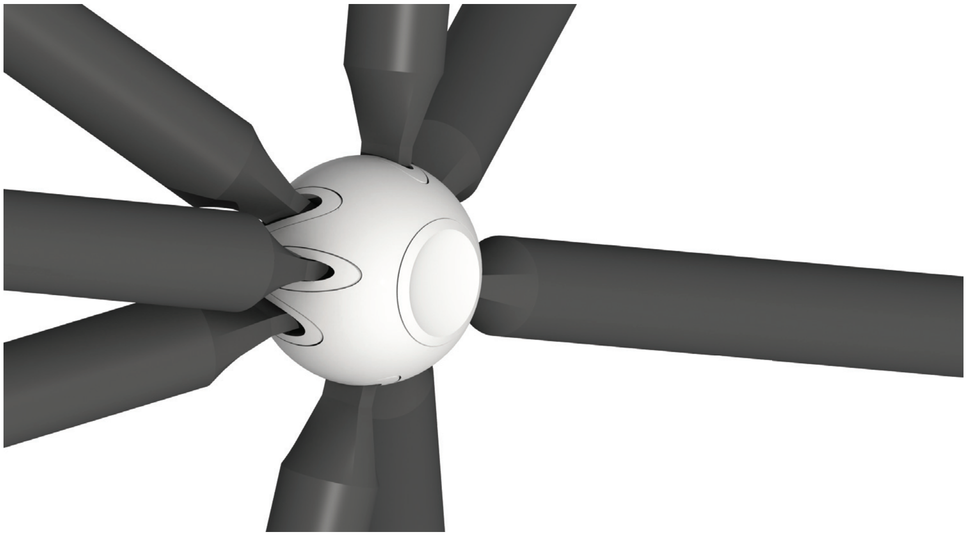


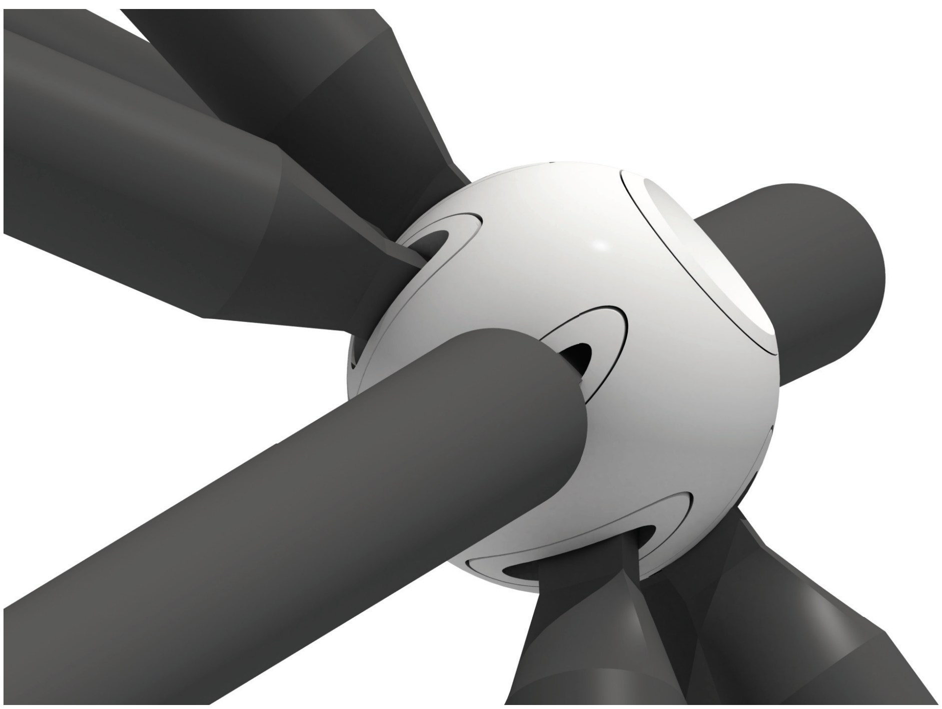




2.44 Plated Joint Details

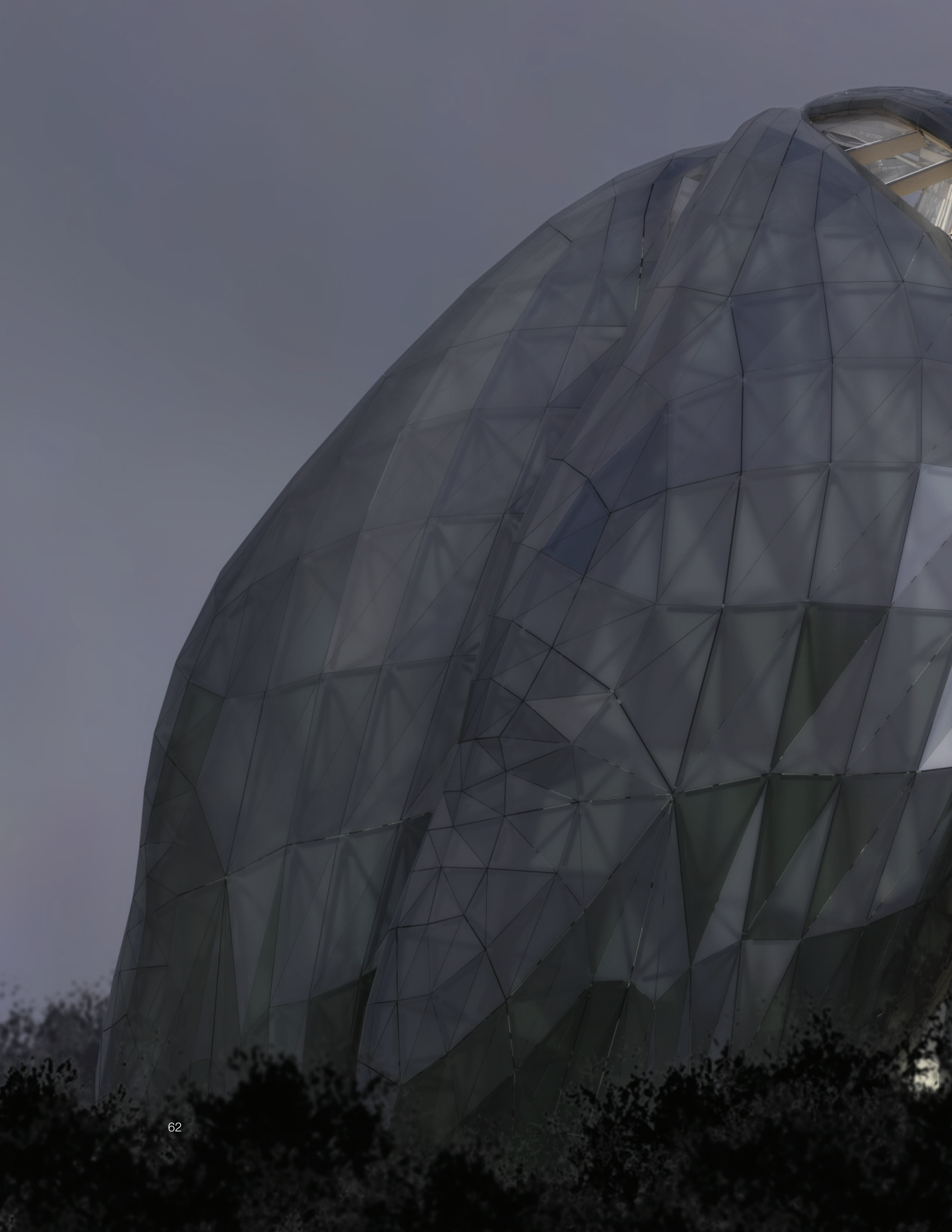
Joint Footprint: 300mm x 300mm
Joint Material: Solid Machined Steel Plates





2.45 Spherical Joint Details

Joint Diameter: 230mm
Joint Material: Solid Machined Steel Bearing



Parametric Processes

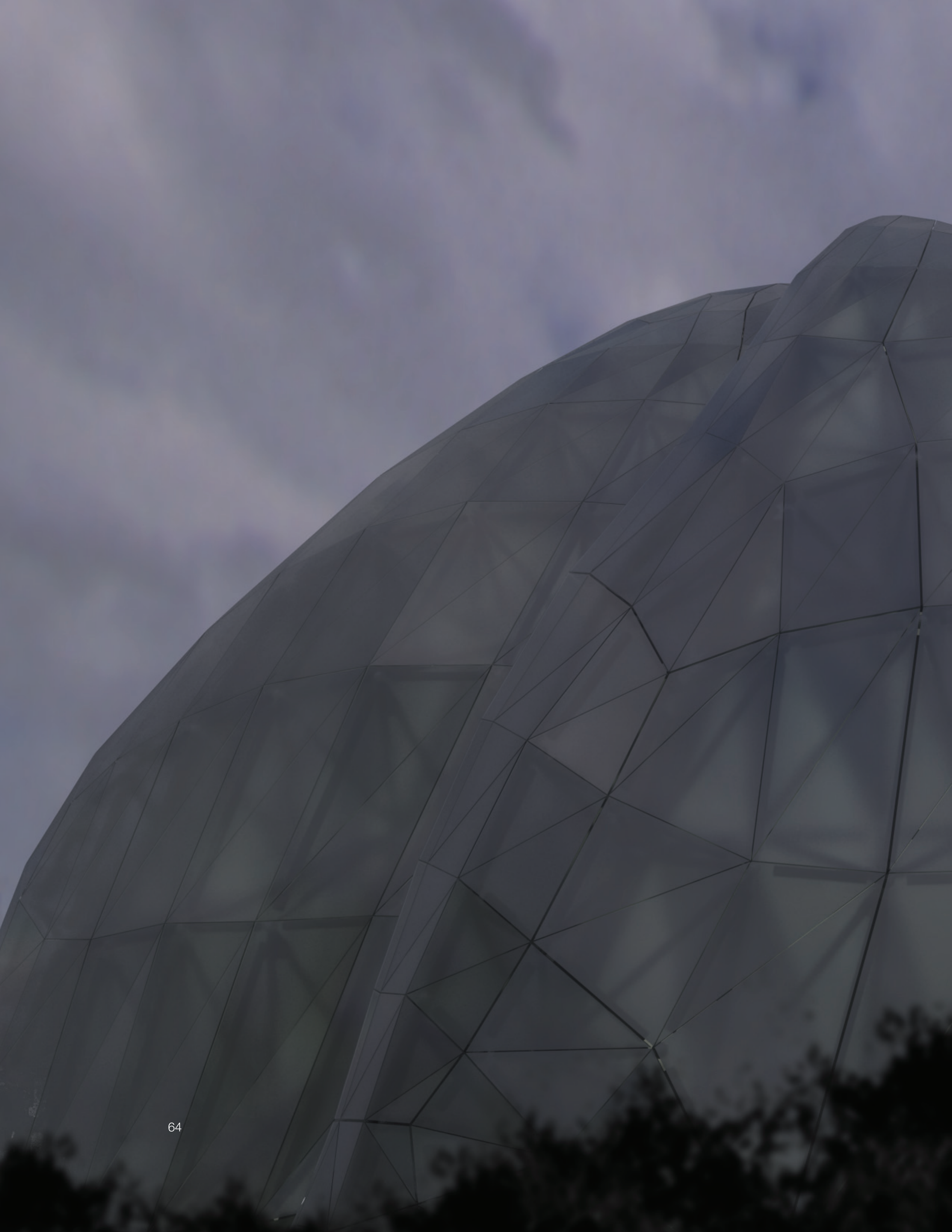
By using a product such as CATIA, Hariri Pontarini were able to build solid models of structure and skin and create scaled samples—milled by a five-axis milling machine—to gain an understanding of how the final component might behave. Its tight integration with manufacturing technologies makes it effective as a professional tool.

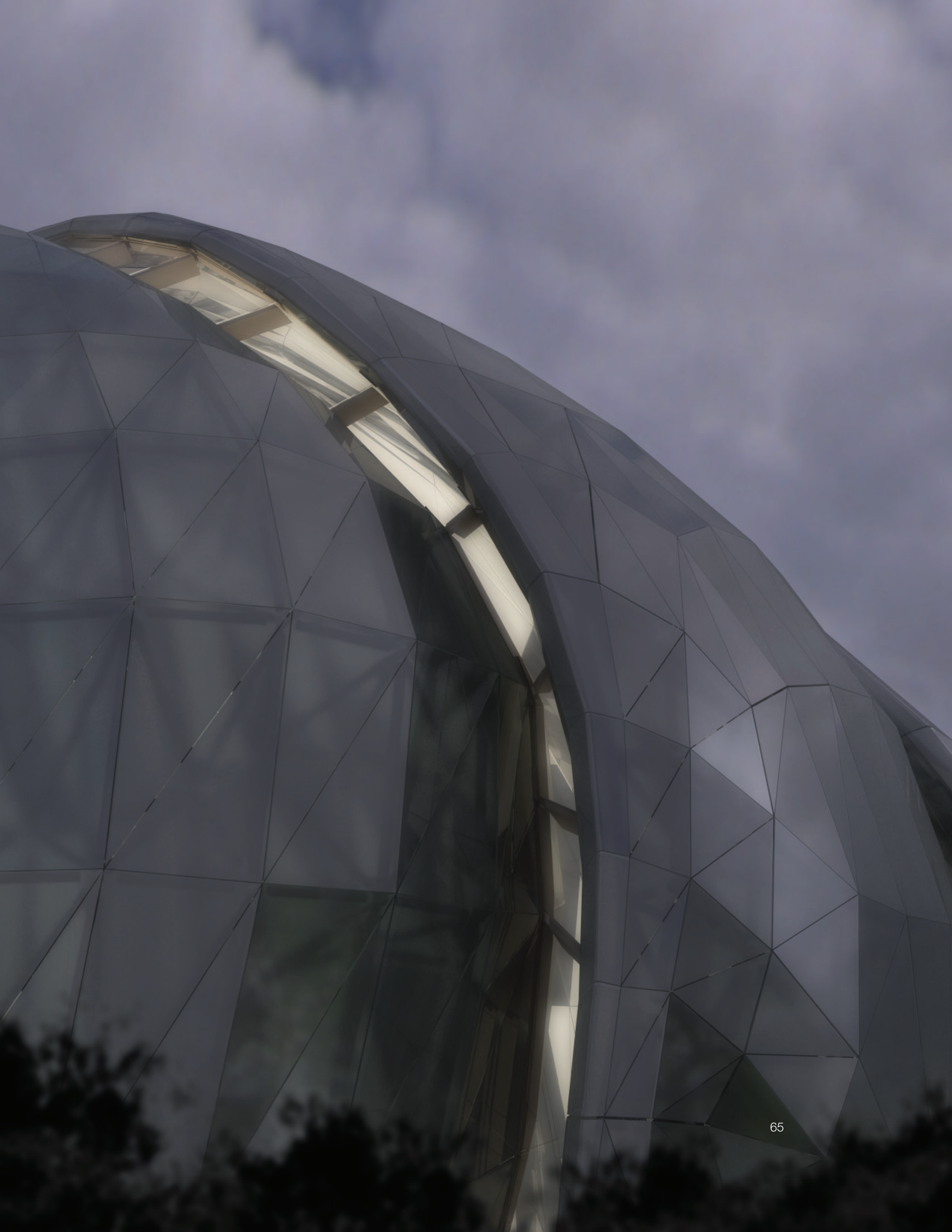
However, a dichotomy between design and construction becomes evident when using this tool. I suggest CATIA is not, in fact, a design tool, it is a manufacturing tool. It has an extremely comprehensive toolset, but tends to require a path of production that leads directly from design to manufacturing with a minimum of revisions.

Generative Components (GC) consists of a set of tools built for conceptual designers, it was not designed to be an engineering and manufacturing application. Although it employs "fabrication planning" components, these components lack the breadth of the manufacturing tools found in CATIA. I believe GC can be categorized as a tool for designers, while CATIA should be viewed as a tool for technologists, engineers and manufacturers.

The sub-components of this chapter were used to evaluate the construction of the Bahá'í Temple model and to gain a better understanding of the building components. The Skin and Structure and Digital Detailing components of this chapter exposed the complexities of the space frame structure through a number of descriptive diagrams. Figure 2.46 reveals the unusual shape of the cladding components, these components have been cut to suit a visual aesthetic and not for the sake of structural optimization. In the next chapter, the Exposition, the surface is not faceted in this way. The Exposition focuses on the use of Generative Components to produce many alternative variations, as opposed to visually-discrete solutions.

LEFT/NEXT *2.46/2.47 The following photo-realistic rendering of the Temple at dusk expresses the true reality of the exterior.*





transaction modelBased "CS"{feature GC.CoordinateSystem CoordinateSystem01{CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem04 { CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (-5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem02 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem03 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (-5); Ztranslation = 0.0; }} transaction modelBased "GC Placement" { feature GC.GraphVariable RAD3 { Value = 3; } feature GC.GraphVariable RAD2 { Value = 3; } feature GC.GraphVariable RAD4 { Value = 3; } feature GC.GraphVariable RAD { Value = 3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { CoordinateSystem01 = CoordinateSystem03; RAD = RAD3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { CoordinateSystem01 = CoordinateSystem02; RAD = RAD2; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { CoordinateSystem01 = CoordinateSystem04; RAD = RAD4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { CoordinateSystem01 = CoordinateSystem01; RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD4 { SymbolXY = {104, 102}; } feature GC.GraphVariable RAD2 { SymbolXY = {102, 102}; } feature GC.GraphVariable RAD { SymbolXY = {101, 102}; } feature GC.GraphVariable RAD3 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 101}; }} transaction modelBased RAD Limit" { deleteFeature RAD4; deleteFeature RAD3; deleteFeature RAD2; feature GC.GraphVariable RAD { UsesNumericLimits = true; NumericHighLimit = 10.0; SymbolXY = {106, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {105, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD { SymbolXY = {106, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 102}; } feature GC.CoordinateSystem baseCS { SymbolXY = {99, 100}; }} transaction modelBased "Upper Line Limit" { feature GC.GraphVariable SYS01_L01 { Value = 7; } feature GC.GraphVariable SYS01_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS01_L03 { Value = 4; } feature GC.GraphVariable L_LIM { Value = RAD*10; } feature GC.GraphVariable SYS01_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS01_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS01_L02 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { L1 = SYS01_L01; L2 = SYS01_L02; L3 = SYS01_L03; RAD1 = SYS01_RAD01; RAD2 = SYS01_RAD02; RAD3 = SYS01_RAD03; SymbolXY = {99, 102}; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS02_L03 { Value = 5; } feature GC.GraphVariable SYS02_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS02_L02 { Value = 7; } feature GC.GraphVariable SYS02_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS02_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS02_L01 { Value = 6; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { L1 = SYS02_L01; L2 = SYS02_L02; L3 = SYS02_L03; RAD1 = SYS02_RAD01; RAD2 = SYS02_RAD02; RAD3 = SYS02_RAD03; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS03_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS03_L01 { Value = 6; } feature GC.GraphVariable SYS03_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS03_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS03_L03 { Value = 3; } feature GC.GraphVariable SYS03_L02 { Value = 4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { L1 = SYS03_L01; L2 = SYS03_L02; L3 = SYS03_L03; RAD1 = SYS03_RAD01; RAD2 = SYS03_RAD02; RAD3 = SYS03_RAD03; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS04_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS04_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS04_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS04_L02 { Value = 6; } feature GC.GraphVariable SYS04_L03 { Value = 8; } feature GC.GraphVariable SYS04_L01 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { L1 = SYS04_L01; L2 = SYS04_L02; L3 = SYS04_L03; RAD1 = SYS04_RAD01; RAD2 = SYS04_RAD02; RAD3 = SYS04_RAD03; }} transaction modelBased "Radian Limits" { transaction modelBased "State at which new feature type, GC._cvGenerator_Multi_001, created" {} transaction modelBased "CS"{feature GC.CoordinateSystem CoordinateSystem01{CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem04 { CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (-5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem02 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem03 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (-5); Ztranslation = 0.0; }} transaction modelBased "GC Placement" { feature GC.GraphVariable RAD3 { Value = 3; } feature GC.GraphVariable RAD2 { Value = 3; } feature GC.GraphVariable RAD4 { Value = 3; } feature GC.GraphVariable RAD { Value = 3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { CoordinateSystem01 = CoordinateSystem03; RAD = RAD3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { CoordinateSystem01 = CoordinateSystem02; RAD = RAD2; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { CoordinateSystem01 = CoordinateSystem04; RAD = RAD4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { CoordinateSystem01 = CoordinateSystem01; RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD4 { SymbolXY = {104, 102}; } feature GC.GraphVariable RAD2 { SymbolXY = {102, 102}; } feature GC.GraphVariable RAD { SymbolXY = {101, 102}; } feature GC.GraphVariable RAD3 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 101}; }} transaction modelBased RAD Limit" { deleteFeature RAD4; deleteFeature RAD3; deleteFeature RAD2; feature GC.GraphVariable RAD { UsesNumericLimits = true; NumericHighLimit = 10.0; SymbolXY = {106, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {105, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD { SymbolXY = {106, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 102}; } feature GC.CoordinateSystem baseCS { SymbolXY = {99, 100}; }} transaction modelBased "Upper Line Limit" { feature GC.GraphVariable SYS01_L01 { Value = 7; } feature GC.GraphVariable SYS01_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS01_L03 { Value = 4; } feature GC.GraphVariable L_LIM { Value = RAD*10; } feature GC.GraphVariable SYS01_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS01_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS01_L02 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { L1 = SYS01_L01; L2 = SYS01_L02; L3 = SYS01_L03; RAD1 = SYS01_RAD01; RAD2 = SYS01_RAD02; RAD3 = SYS01_RAD03; SymbolXY = {99, 102}; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS02_L03 { Value = 5; } feature GC.GraphVariable SYS02_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS02_L02 { Value = 7; } feature GC.GraphVariable SYS02_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS02_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS02_L01 { Value = 6; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { L1 = SYS02_L01; L2 = SYS02_L02; L3 = SYS02_L03; RAD1 = SYS02_RAD01; RAD2 = SYS02_RAD02; RAD3 = SYS02_RAD03; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS03_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS03_L01 { Value = 6; } feature GC.GraphVariable SYS03_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS03_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS03_L03 { Value = 3; } feature GC.GraphVariable SYS03_L02 { Value = 4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { L1 = SYS03_L01; L2 = SYS03_L02; L3 = SYS03_L03; RAD1 = SYS03_RAD01; RAD2 = SYS03_RAD02; RAD3 = SYS03_RAD03; }}transaction modelBased "More GV" { feature GC.GraphVariable SYS04_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS04_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS04_RAD03 { Value = 0.8; }feature GC.GraphVariable SYS04_L02 { Value = 6; } feature GC.GraphVariable SYS04_L03 { Value = 8; } feature GC.GraphVariable SYS04_L01 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { L1 = SYS04_L01; L2 = SYS04_L02; L3 = SYS04_L03; RAD1 = SYS04_RAD01; RAD2 = SYS04_RAD02; RAD3 = SYS04_RAD03; }} transaction modelBased "Radian Limits" { transaction modelBased "State at which new feature type, GC._cvGenerator_Multi_002, created" {} transaction modelBased "CS"{feature GC.CoordinateSystem CoordinateSystem01{CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem04 { CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (-5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem02 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem03 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (-5); Ztranslation = 0.0; }} transaction modelBased "GC Placement" { feature GC.GraphVariable RAD3 { Value = 3; } feature GC.GraphVariable RAD2 { Value = 3; } feature GC.GraphVariable RAD4 { Value = 3; } feature GC.GraphVariable RAD { Value = 3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { CoordinateSystem01 = CoordinateSystem03; RAD = RAD3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { CoordinateSystem01 = CoordinateSystem02; RAD = RAD2; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { CoordinateSystem01 = CoordinateSystem04; RAD = RAD4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { CoordinateSystem01 = CoordinateSystem01; RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD4 { SymbolXY = {104, 102}; } feature GC.GraphVariable RAD2 { SymbolXY = {102, 102}; } feature GC.GraphVariable RAD { SymbolXY = {101, 102}; } feature GC.GraphVariable RAD3 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 101}; }} transaction modelBased RAD Limit" { deleteFeature RAD4; deleteFeature RAD3; deleteFeature RAD2; feature GC.GraphVariable RAD { UsesNumericLimits = true; NumericHighLimit = 10.0; SymbolXY = {106, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { RAD = RAD; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {105, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD { SymbolXY = {106, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 102}; } feature GC.CoordinateSystem baseCS { SymbolXY = {99, 100}; }} transaction modelBased "Upper Line Limit" { feature GC.GraphVariable SYS01_L01 { Value = 7; } feature GC.GraphVariable SYS01_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS01_L03 { Value = 4; } feature GC.GraphVariable L_LIM { Value = RAD*10; } feature GC.GraphVariable SYS01_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS01_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS01_L02 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { L1 = SYS01_L01; L2 = SYS01_L02; L3 = SYS01_L03; RAD1 = SYS01_RAD01; RAD2 = SYS01_RAD02; RAD3 = SYS01_RAD03; SymbolXY = {99, 102}; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS02_L03 { Value = 5; } feature GC.GraphVariable SYS02_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS02_L02 { Value = 7; } feature GC.GraphVariable SYS02_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS02_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS02_L01 { Value = 6; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { L1 = SYS02_L01; L2 = SYS02_L02; L3 = SYS02_L03; RAD1 = SYS02_RAD01; RAD2 = SYS02_RAD02; RAD3 = SYS02_RAD03; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS03_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS03_L01 { Value = 6; } feature GC.GraphVariable SYS03_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS03_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS03_L03 { Value = 3; } feature GC.GraphVariable SYS03_L02 { Value = 4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { L1 = SYS03_L01; L2 = SYS03_L02; L3 = SYS03_L03; RAD1 = SYS03_RAD01; RAD2 = SYS03_RAD02; RAD3 = SYS03_RAD03; }} transaction modelBased "More GV" { feature GC.GraphVariable SYS04_RAD01 { Value = 0.1; } feature GC.GraphVariable SYS04_RAD02 { Value = 0.5; } feature GC.GraphVariable SYS04_RAD03 { Value = 0.8; } feature GC.GraphVariable SYS04_L02 { Value = 6; } feature GC.GraphVariable SYS04_L03 { Value = 8; } feature GC.GraphVariable SYS04_L01 { Value = 5; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { L1 = SYS04_L01; L2 = SYS04_L02; L3 = SYS04_L03; RAD1 = SYS04_RAD01; RAD2 = SYS04_RAD02; RAD3 = SYS04_RAD03; }} transaction modelBased "Radian Limits" { transaction modelBased "State at which new feature type, GC._cvGenerator_Multi_003, created" {} transaction modelBased "CS"{feature GC.CoordinateSystem CoordinateSystem01{CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem04 { CoordinateSystem = baseCS; Xtranslation = <free> (5); Ytranslation = <free> (-5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem02 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (5); Ztranslation = 0.0; } feature GC.CoordinateSystem CoordinateSystem03 { CoordinateSystem = baseCS; Xtranslation = <free> (-5); Ytranslation = <free> (-5); Ztranslation = 0.0; }} transaction modelBased "GC Placement" { feature GC.GraphVariable RAD3 { Value = 3; } feature GC.GraphVariable RAD2 { Value = 3; } feature GC. phVariable RAD4 { Value = 3; } feature GC.GraphVariable RAD { Value = 3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { CoordinateSystem01 = CoordinateSystem03; RAD = RAD3; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { CoordinateSystem01 = CoordinateSystem02; RAD = RAD2; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { CoordinateSystem01 = CoordinateSystem04; RAD = RAD4; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { CoordinateSystem01 = CoordinateSystem01; RAD = RAD; }} transaction modelBased "Mod Graph" { feature GC.GraphVariable RAD4 { SymbolXY = {104, 102}; } feature GC.GraphVariable RAD2 { SymbolXY = {102, 102}; } feature GC.GraphVariable RAD { SymbolXY = {101, 102}; } feature GC.GraphVariable RAD3 { SymbolXY = {103, 102}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00104 { SymbolXY = {104, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00102 { SymbolXY = {102, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00101 { SymbolXY = {101, 101}; } feature GC._cvGenerator_Single_001 _cvGenerator_Single_00103 { SymbolXY = {103, 101}; }} transaction modelBased RAD Limit" { deleteFeature RAD4; deleteFeature RAD3; deleteFeature RAD2; feature GC.GraphVariable RAD { UsesNumericLimits = true; NumericHighLimit = 10.0; SymbolXY = {106, 101}; } feature

C | EXPOSITION

ENGINEERING A TOOL FOR DESIGN

CHAPTER TITLE ILLUSTRATION *This illustration reveals the contents of the Generative Components Transaction file that was used to build a comprehensive mock-up of the Temple Fin.*

The Exposition section of this thesis uses the contents of Bahá'í Temple as a tool to explore and evaluate Bentley's Generative Components (GC). GC is a parametric design tool that facilitates the creation and extraction of design concepts through virtual modeling or scripting. GC has four primary workspaces (see 3.08), *Transaction*, *Model*, *Symbolic* and *Script*. The *Transaction* workspace records changes in the state of the model. This is a user initiated process. *Transaction* steps are added automatically when a user generates a *Feature*. The *Modeling* workspace is where parametric models are built; it contains tools that are visually accessible through a tab in the *Transaction* workspace. The *Symbolic* workspace is where the *Symbolic Graph* resides. The graph provides a visual diagram of the relationship between the *Features* in the model (see 3.03). The *Scripting* workspace is also accessible through an icon within the *Transaction* workspace. The *Script* contains all of the information necessary to build a model within GC (see 3.02). If you were to copy and paste compatible code from a text document into the *Script Editor*, GC would construct a model from its contents. Precise material that relates to the definition of terms used in the software can be found in the content following this introduction. Documentation and tutorials can be found in the form of texts and example files available in print from Bentley Systems[1] and online from the Smart Geometry Design Science website, hosted by the Canadian Research and Design Network.[2]

This work examines components of a new tool for design. An introduction to the tool frames its features: the tool is divided into manageable components to evaluate aspects that address its flexibility, breadth of features and integration with other technologies. The application of this tool to the Temple geometry is also divided into several sub-components. The geometry is separated by virtue of its architectural components: the overall formal gesture (the form), interior cladding, exterior cladding and structure. Each of the Temple components is accompanied by an explicit description of the application of Generative Components. I include several customized features that I have developed (see 3.11 and 3.12). This chapter is concludes with an assessment of the state of Generative Components and possible areas of future research and development.

About the Exposition

Designers will use digital tools in manners other than which they were intended. Typically they are used for the pre or post rationalization of a conceptual idea. When we design or engineer architecture a large part of the process involves mass customization. To save time and money we limit the amount of differentiation found in details, building components and building features; this adds clarity to the conceptual aspect of the design, reduces overhead and streamlines the design process. *Streamlined components* from each project become part of a larger database of architectural solutions. This is a *scalar process* that adds value at many levels.

Trimming budgets and streamlining project details are obvious uses for tools such as CATIA and Generative Components. Architect Lucien Kroll wrote, "The computerized architect should surely not be limited to what can be run through his mill."[3] The richness of such tools should be exploited and altered through their application.

Features and Functions

Generative Components has two major libraries which contain the bulk of its functionality. A *library* is defined as a collection of items that have been made available to the user to help facilitate the use of the tool.

Generative Components' major libraries are composed of *Features* and *Functions* (also referred to as *Global Functions*). *Features* are objects that aid in the creation of complex systems. For example: points, lines, arcs and shapes are all *Features*.[4] GC contains an extensive list of base *Features*, but also allows the user to generate his own *Features*. These are known as *Components*, hence the term Generative Components. *Functions* are far more complex. *Functions* are predefined equations that process geometric or mathematical data (see 3.15). *Functions* can collect data which can be used to drive discrete variables within *Components*. For example, the conceptual form of the GC Temple is based on a set of concentric circles. The radius of the circles is determined by a *Function* which uses an exponent that increases by an increment of *n*. This variable is generated by a formula that does not allow linear progression and therefore the size of the circles will never increase at a linear rate. *Functions* available to the user include the ability to generate and return lists, the ability to evaluate and return a value from GC *Script* expressions, the ability to call *Functions* from within *Functions*, and the ability to identify the square root of a value. GC has an extensive list of approximately eighty *Functions*. A *Graph Function* is a *Feature* that facilitates the creation of customized *Functions*. Customized *Functions* can be used in the event that existing *Functions* do not satisfy the functionality required.

These software components are at the heart of the Generative Components system. The section entitled Programmable Structure provides a broader context of additional modeling tools available within Generative Components.

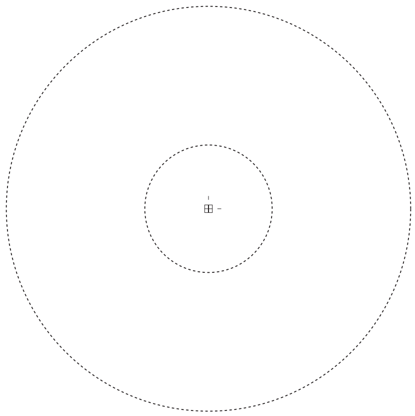
The Economies of a Flexible Design Tool

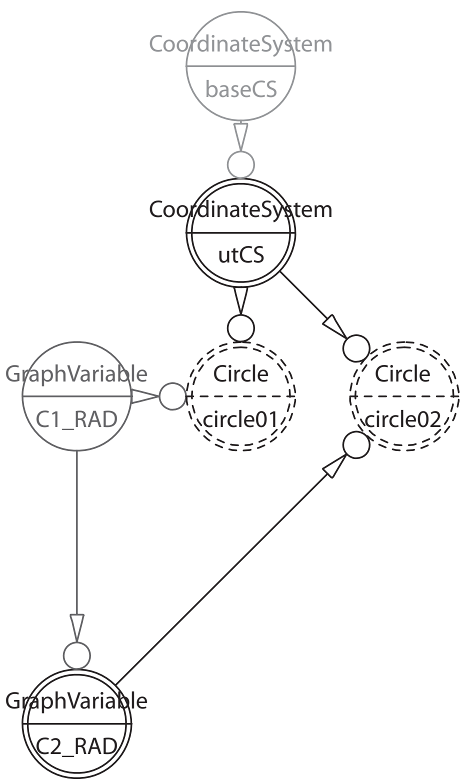
Defining the fundamental principles of Generative Components positions the content of this chapter within the pretext of the process. The *platform* in which GC was developed plays a significant role in this evaluation and effects the process based on the premise of *robustness* and process driven performance. The conceptual process behind the workflow of this tool is what makes it entirely unique, and at times quite difficult to grasp.

Generative Components exists in a paradox, one which is bound by environmental computing constraints and yet unbound by the aspirations of the designer who commands it. It creates proficient results but requires unfathomable amounts of resources to do this responsively. Content created in Generative Components is represented by three elegant models; *Script Transactions*, *Symbolic Graphs* and model geometry (see 3.01-3.07). The *physical geometry* of a visual or three dimensional model is only created when the user tells the application to do so. This reduces the amount of permanent storage and allows the user to edit the *internal code* outside of the context of the platform. Unfortunately structuring the system in this way means that every change or modification that is added to the *transaction file* puts a strain on computing resources. Although it may seem more constrained than flexible, the first version of GC is simply a solid basis for future revisions. This software was developed on an advanced, modern and *object oriented* platform; one that is extensible, open and customizable.

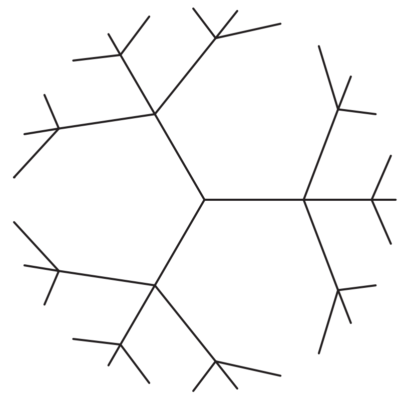
Generative Components is a parametric design tool that facilitates the drawing or extraction of a design concept. It can be initiated as an *active or passive modeling environment*. The content that GC draws is parametric. Changing *pre-scripted elements* of a transaction propagates change throughout the file, which dynamically updates the original script. The developers of Generative Components define it as "an application packed full of ideas,"[5] they go on to state that these ideas "are based on concepts drawn from computational geometry, design composition and procedural and declarative computer languages. These concepts are standard within their respective domains, but may be unfamiliar to designers in practice."[6] Dr. Aish addresses the need for such a tool in a Generative Components training document, "The Generative Components system has the potential to span the architectural process from concept formation to digital fabrication in a system of related design models.

RIGHT (CLOCKWISE FROM TOP LEFT) *3.01 Geometry, 3.02 Sample of Generative Components Transaction code, 3.03 Symbolic Graph.*

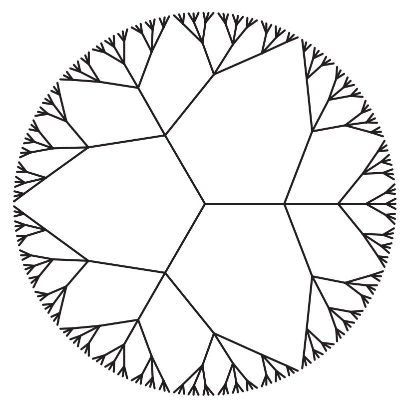


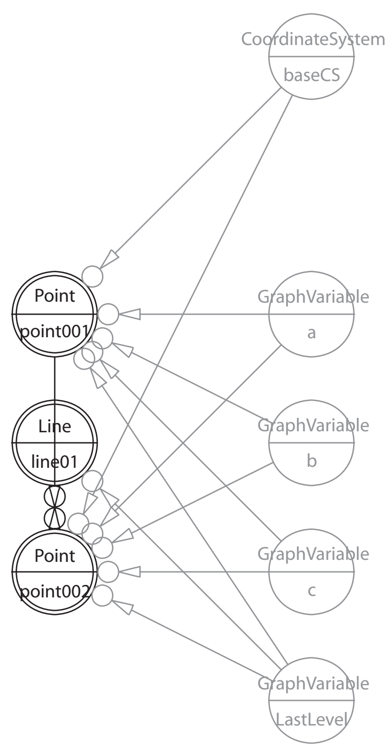



```
transaction modelBased "C2"
{
    feature GC.CoordinateSystem utCS
    {
        SymbolXY                = {100, 101};
    }
    feature GC.GraphVariable C1_RAD
    {
        UsesNumericLimits       = true;
        NumericLowLimit         = 1.0;
        NumericHighLimit        = 30.0;
        SymbolXY                = {99, 103};
    }
    feature GC.Circle circle01
    {
        SymbolXY                = {100, 103};
        Construction            = ConstructionOption.Construction;
    }
    feature GC.GraphVariable C2_RAD
    {
        Value                   = C1_RAD*Pow(2, C1_RAD/3);
        SymbolXY                = {99, 105};
    }
    feature GC.Circle circle02
    {
        CenterPoint             = utCS;
        Radius                  = C2_RAD;
        Support                 = utCS.XYplane;
        Construction            = ConstructionOption.Construction;
    }
}
```



```
transaction modelBased “Create points”
{
    feature GC.Point point0001
    {
        Function = function(cs,a,b,c,LastLevel)
                        {
                        DPoint3d points = {};
                        value LastPointOnLevel=0;
                        value Rotation=0.0;
                        value Scale=0.0;
                        for(int Level = 0; Level <= LastLevel; Level++)
                        {
                        if(Level != 0)
                        {
                        LastPointOnLevel=3*LastPointOnLevel+2;
                         Rotation=Rotation+360.0/(LastPointOnLevel+1);
                        Scale=Scale+1.0/Level;
                        }
                        for(int PointOnLevel=0;PointOnLevel<=LastPointOnLevel;PointOnLevel++)
                        {
                        Point points1 = CreateChildFeature(“Point”,this);
                        value x=Scale*a*Cos(360.0*PointOnLevel/(LastPointOnLevel+1)-Rotation);
                        value y=Scale*b*Sin(360.0*PointOnLevel/(LastPointOnLevel+1)-Rotation);
                        value z=Scale*c;
                        points1.ByCartesianCoordinates(baseCS, x, y, z);
                        }
                        }
                        };
        FunctionArguments = {baseCS,a,b,c,LastLevel};
    }
}
```

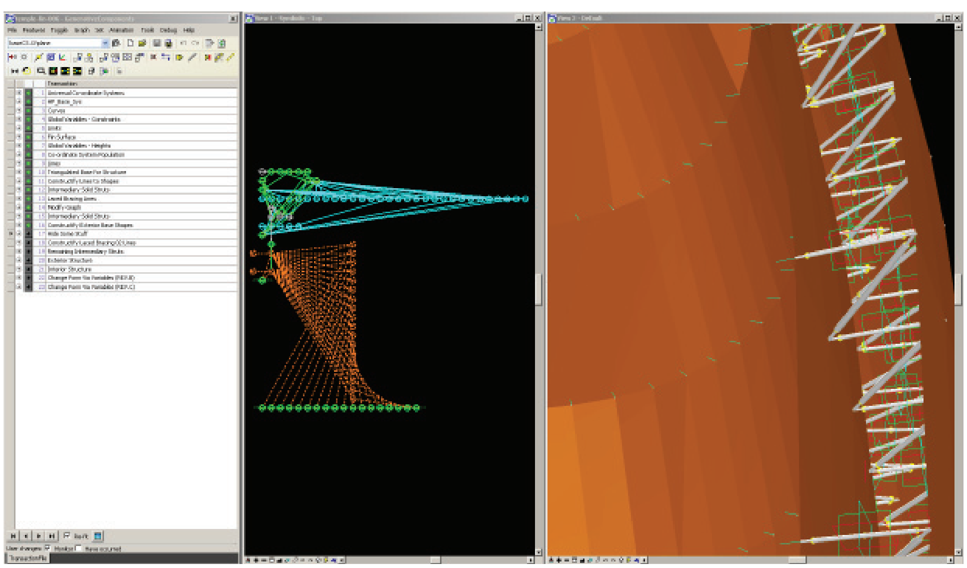




As a design tool for exploratory architecture it also addresses the need for designers to test and confirm the practicality of such exploration."[7]

This chapter focuses on the application of Generative Components to the Bahá'í Temple—a fundamental introduction to the tool is presented to reveal the underlying principles and effects of parametric tools in architectural process. The use of this software demands a certain level of preemptive thought on the behalf of the designer. Every aspect of the content one generates with Generative Components is calculated. The tool is designed in such a way that it requires you to do so. Whether the tool should require you to work in this manner is beyond the scope of this exposition. The topic that is addressed here is how the tool affects the way we work with design tools, not why the tool suggests we work in this manner—this is an area of great breadth and will be categorized in the conclusion of this thesis as an area of future research.

LEFT (CLOCKWISE FROM LEFT) *3.04 Sample of GC Script, 3.05 Model, 3.06 Model with modified script, 3.07 Symbolic Graph.*



3.08 Screen shot of the Generative Components user interface (the Transaction list appears on the left, the Symbolic Model in the centre, followed by the modeling workspace to the right

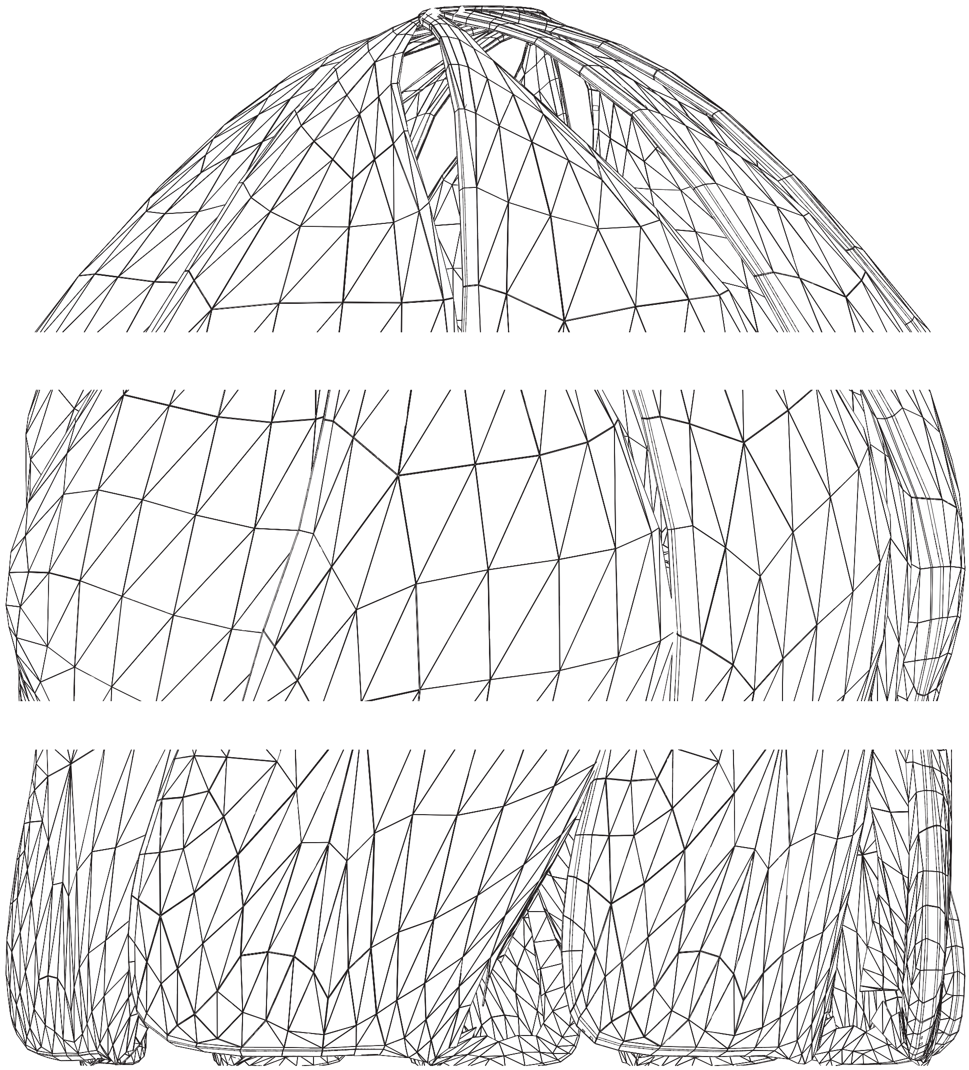
A Generative Bahá'í Temple

The following study divides the Temple into several manageable *Components*. The Temple Fins are the primary focus of this exercise and are divided into the layers discussed in the chapter entitled Application. The underlying *Component* based system is contingent upon a series of concentric circles that increase exponentially. The following sketches (see 3.10 and 3.11) describe the preliminary conceptual model of the *Generative Component*. The rings provide a basis for the overall system which is simply an array of the *Generative Component*.

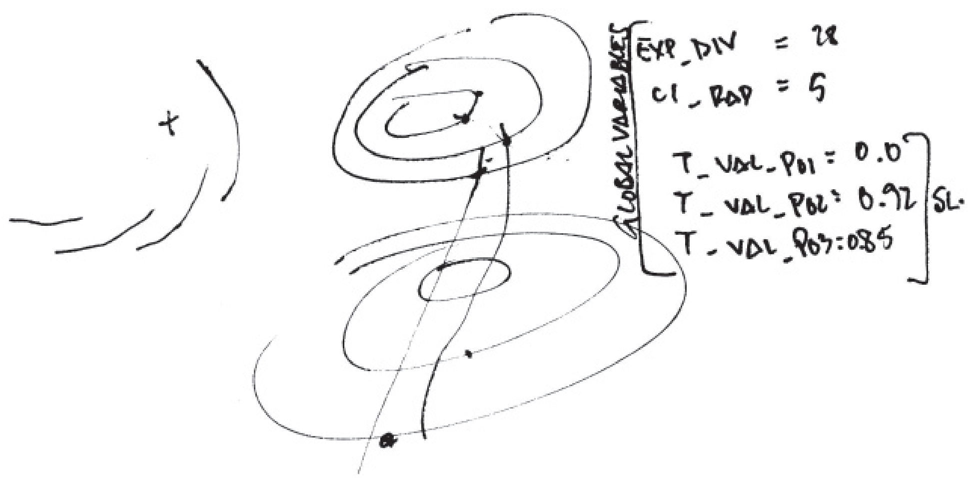
Illustrations in figure 3.12 reveal the mechanics of the base *Generative Component*. The *expressions* define the circles relative to one another and ensure that they are not direct offsets. Exponential growth adds *tri-axial dynamism*; this variable also administers compelling form. The percent of exponential incrementation is controlled by an exponential divider. Decreasing this variable amplifies the distance between circles; increasing this value brings the circles closer to one another. The circles are meaningless in the absence of context. They are simply the base element for the *Generative Component*.

A point is locked to each circle and is manipulated by a *controller* that allows the modeler to adjust its position along the curve in *radians*. The points act as nodes that determine the shape of the curve that is generated by interconnecting them. This approach was established through a study that plotted nodes along key points of the Temple Fin geometry (see 3.13). This data returned a visual representation of curvature that was used to establish the basis—along with the conceptual sketch—for the Generative Components model.

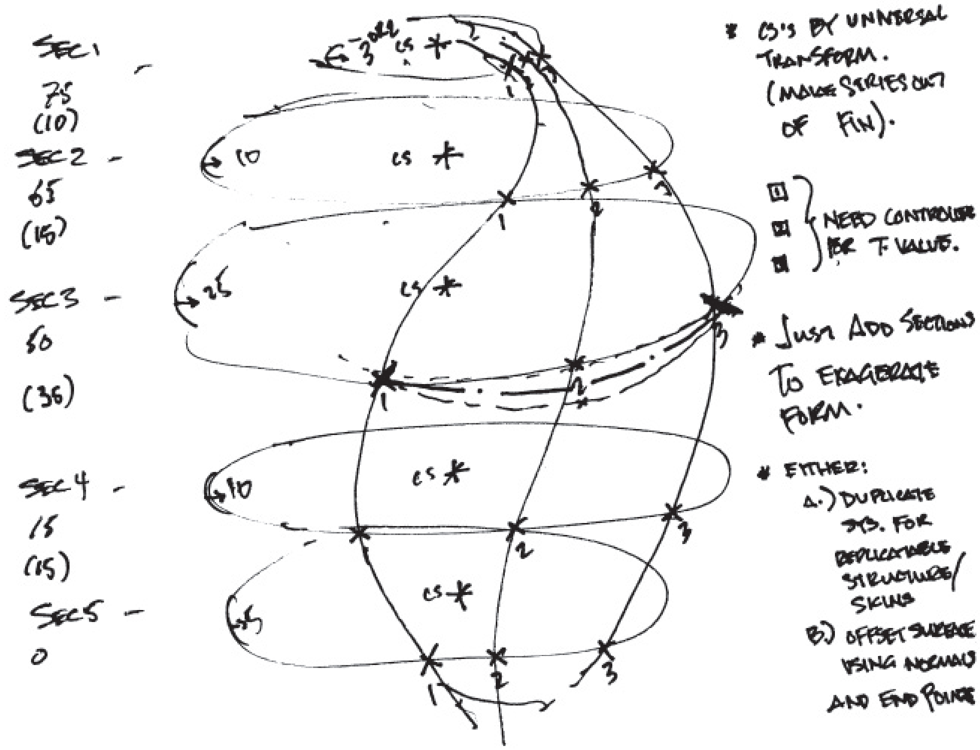
The triangulated alabster cladding and cast-glazing are offset from the base *Component* system used to establish the Generative Temple's form. The space frame structure and iron ball joints are built upon the base *Component* system itself. After a foundation has been established, careful inspection and detail are required to expand components of structure, panelization of stone and glass, and overall form. The physical building features are transposed and used to establish the basis for the GC modeling *Features*.



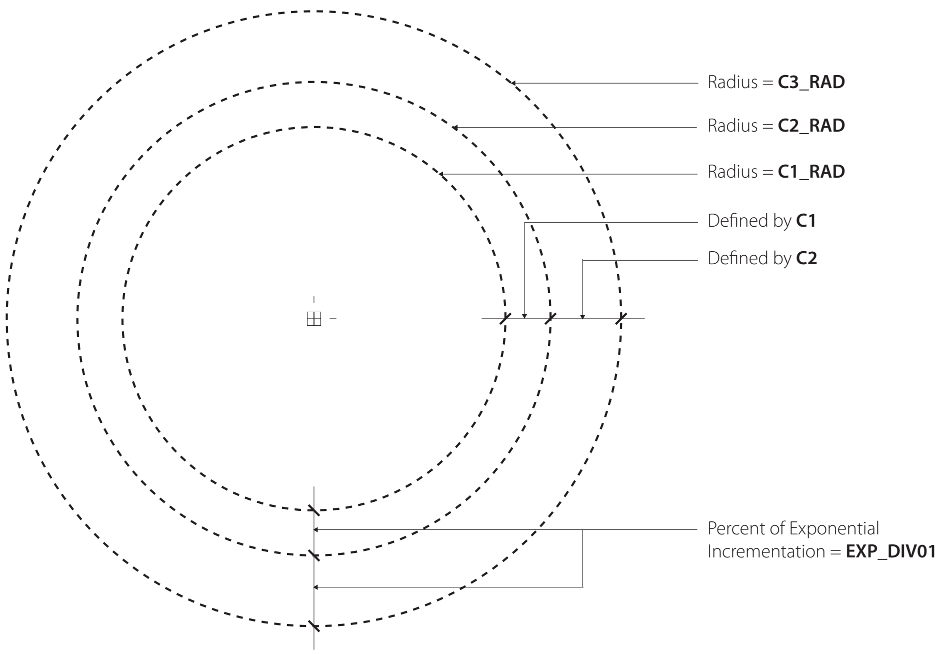
3.09 Detail representing fin curvature and facetization (partial interior and exterior)

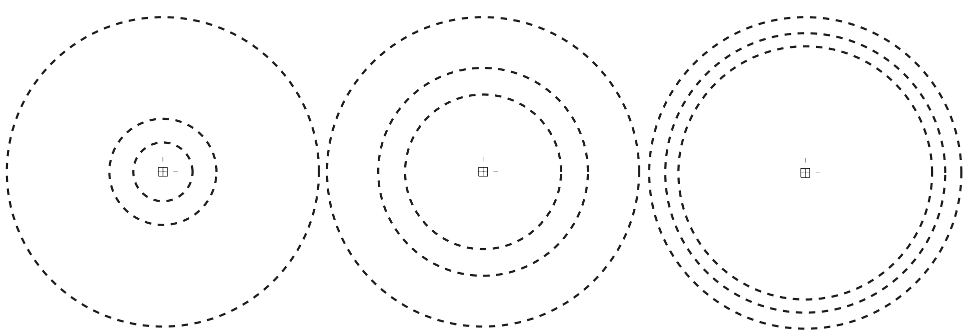


3.10 This sketch represents the conceptual design for the base Component—preliminary variables and sketch diagrams showing the concentric rings are drawn up to define the base system

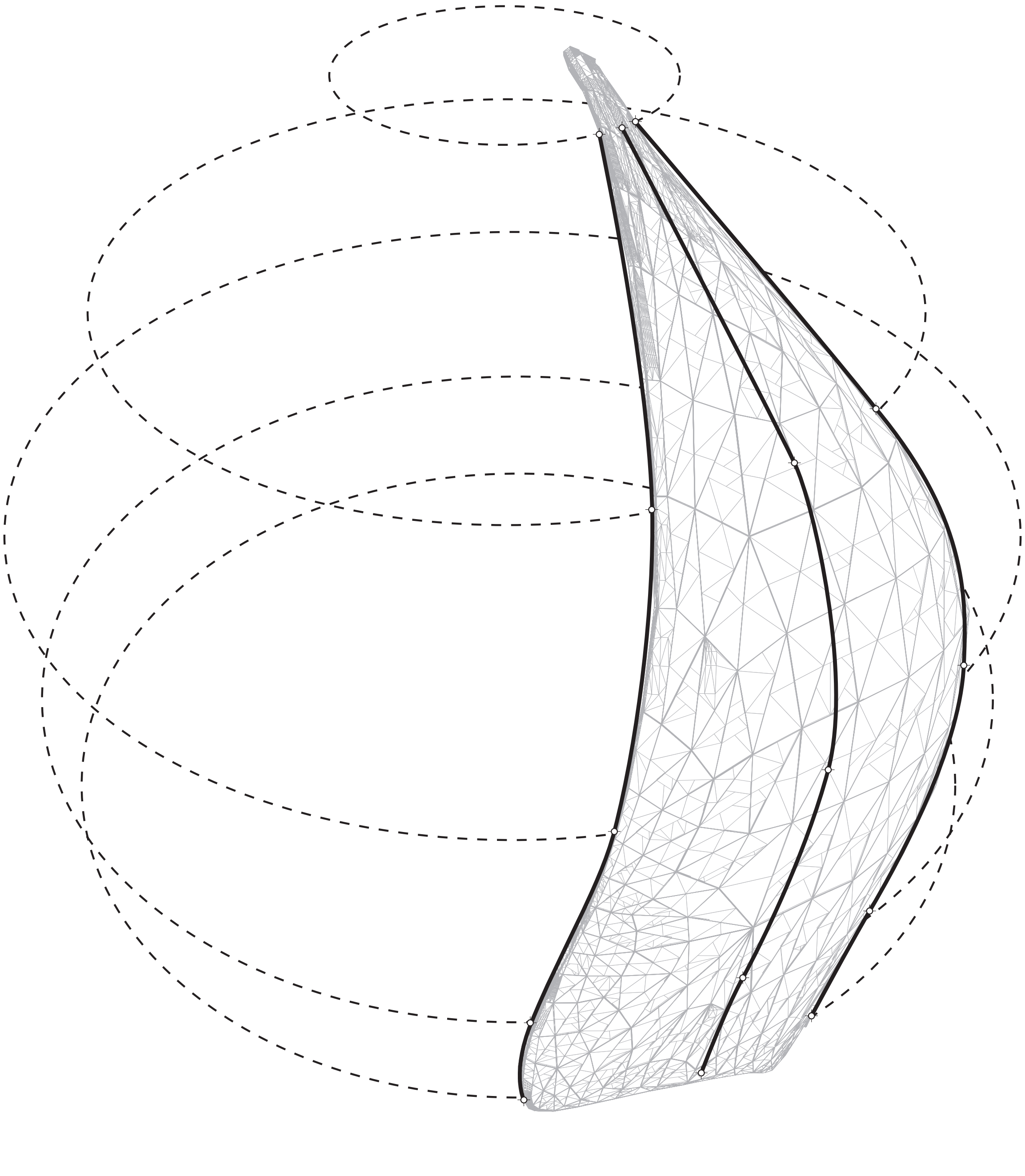


3.11 This illustration reveals the association between arrayed base Components—points on each Component of rings are connected to form a network of curves, the curves are then used to create a surface

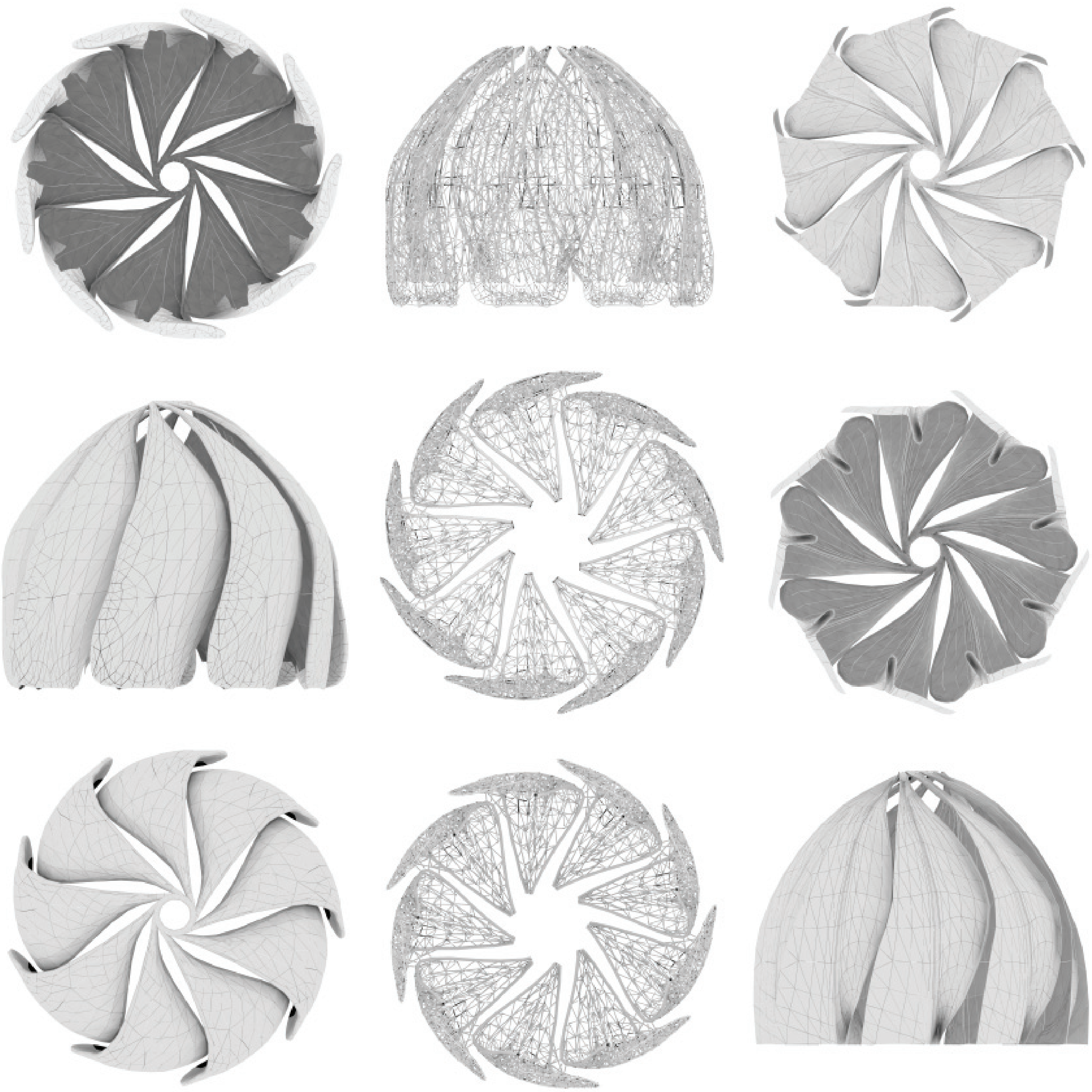




3.12 TOP TO BOTTOM: Cx_RAD variable equations; Variable connections; Base Component with varying values for EXP_DIV01—These illustrations document the construction of the base Component along with any associated variables



3.13 The Generative Component point study reveals the location of the key points used to reconstruct the Fins



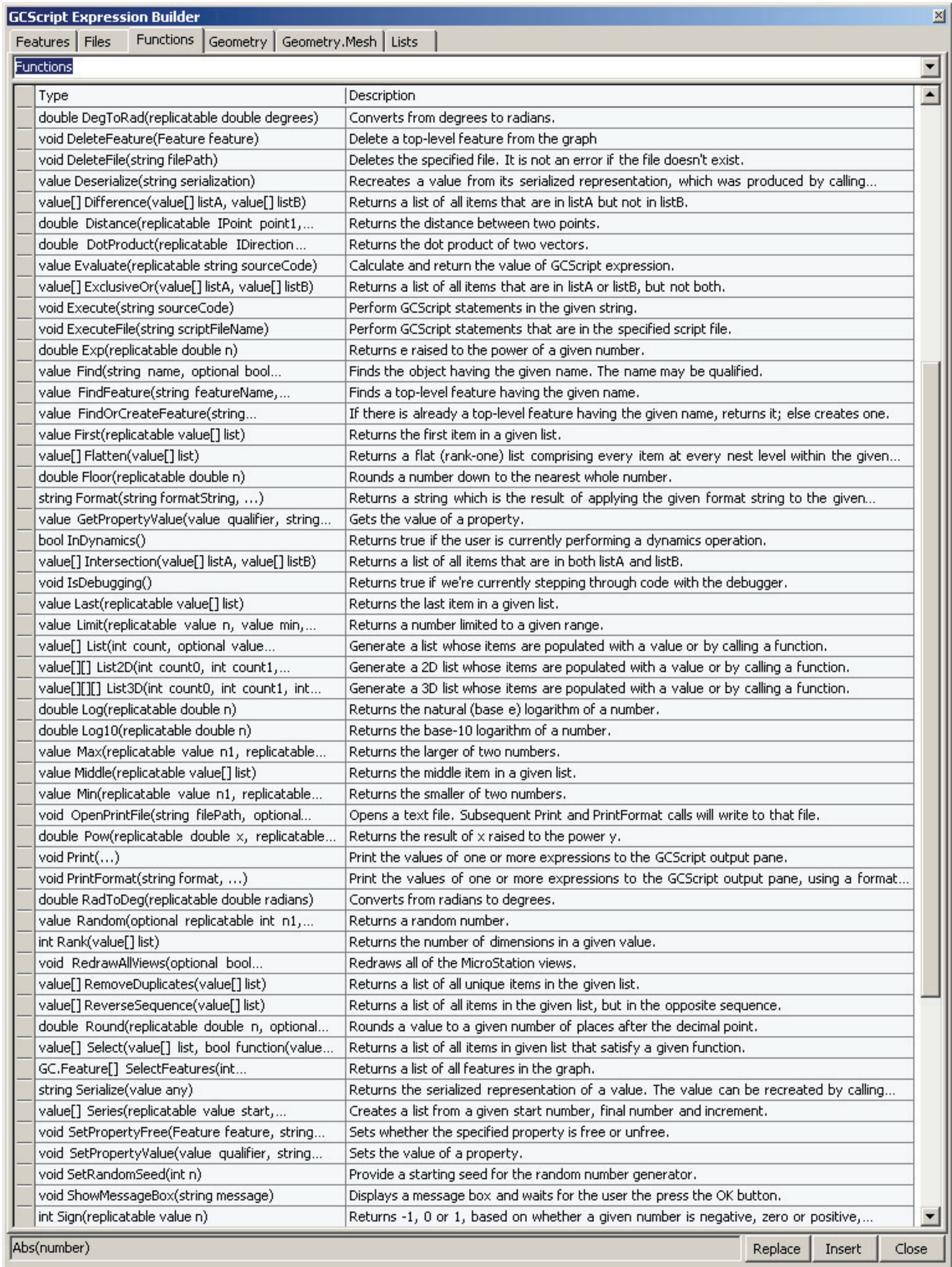
3.14 TOP TO BOTTOM: Cast Glass (bottom), Structure (front), Alabaster (top); Cast Glass (front), Structure (top), Alabaster (bottom); Cast Glass (top), Structure (bottom), Alabaster (front)—these illustrations reveal the complexity in the formal structure of the Fins

PROGRAMMABLE STRUCTURE

The term programmable structure refers to a three dimensional model capable of sending and receiving information (in the form of variables) which has the ability to alter form. This does not include physical structures that instill the mechanics of animation. This sub-section exposes some common control features embedded within GC, namely *Functions* and *Variables*. Here the Temple Components are combined to create a visual likeness of their CATIA counterparts. The connections between components are exposed through a descriptive analysis of the Temple Components.

Generative Components uses several methods which allow the designer to facilitate the transfer of bidirectional data. The most common is a *Graph Variable*. *Graph Variables* act as virtual drivers; they consist of mathematical or code driven initiators that link to, and influence, geometric alteration. The less typical sibling of the *Graph Variable* is the *Graph Function*; a *Graph Function* is far more robust and consists of *object oriented* programming code. Creating loops, conditional statements and automated geometry (as seen in 3.05 and 3.06) are just some of the strengths of a *Graph Function*. Unfortunately the majority of the scripts written in *Graph Functions* force the user to relinquish visual control of an object. Although this can be useful if the designer wishes to minimize graphical interaction or intervention. A competent programmer uses *Graph Functions* as a means of adding *Features* to the functionality of Generative Components, whereas *Graph Variables* are readily accessible and user friendly. *Functions* are easily accessible through the interface which reduces the added overhead of having to memorize syntax (see 3.15). *Functions* alter *Components* through their application of programming based modifiers. A *Function* is an indispensable tool.

The following work embeds this functionality throughout the three-dimensional modeling process. Utilizing this functionality reduces the chances of becoming lost in the complexity of geometry and makes the model easier to interpret by individuals not involved in the creation of the model. Ultimately the content created needs to be understood by a team of designers and not only the creator. An essential phase of this process is embedding the documentation of progression; this allows others to understand the construction of your *Component*.



GCScript Expression Builder

Features | Files | Functions | Geometry | Geometry.Mesh | Lists

Functions

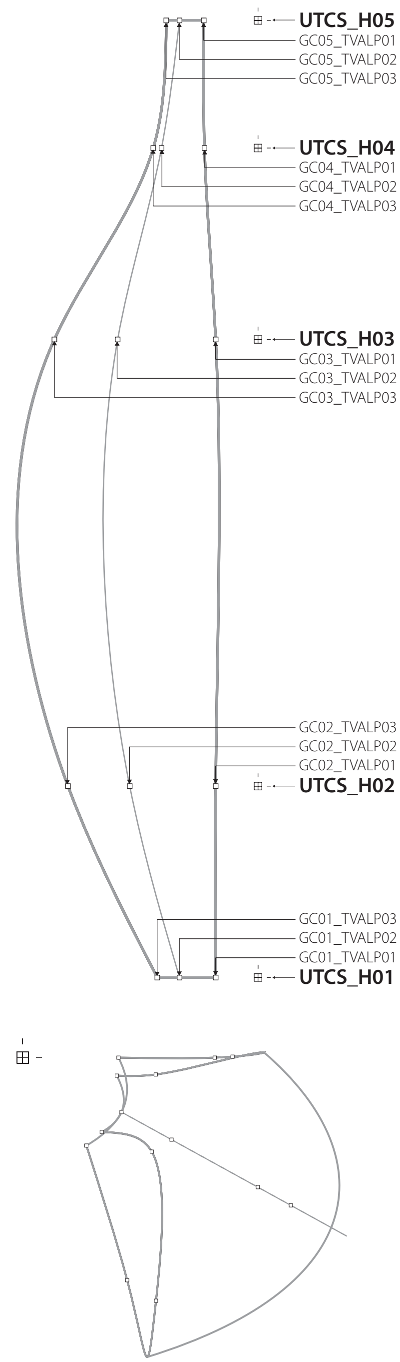
Type	Description
double DegToRad(replicatable double degrees)	Converts from degrees to radians.
void DeleteFeature(Feature feature)	Delete a top-level feature from the graph
void DeleteFile(string filePath)	Deletes the specified file. It is not an error if the file doesn't exist.
value Deserialize(string serialization)	Recreates a value from its serialized representation, which was produced by calling...
value[] Difference(value[] listA, value[] listB)	Returns a list of all items that are in listA but not in listB.
double Distance(replicatable IPoint point1,...	Returns the distance between two points.
double DotProduct(replicatable IDirection...	Returns the dot product of two vectors.
value Evaluate(replicatable string sourceCode)	Calculate and return the value of GCScript expression.
value[] ExclusiveOr(value[] listA, value[] listB)	Returns a list of all items that are in listA or listB, but not both.
void Execute(string sourceCode)	Perform GCScript statements in the given string.
void ExecuteFile(string scriptFileName)	Perform GCScript statements that are in the specified script file.
double Exp(replicatable double n)	Returns e raised to the power of a given number.
value Find(string name, optional bool...	Finds the object having the given name. The name may be qualified.
value FindFeature(string featureName,...	Finds a top-level feature having the given name.
value FindOrCreateFeature(string...	If there is already a top-level feature having the given name, returns it; else creates one.
value First(replicatable value[] list)	Returns the first item in a given list.
value[] Flatten(value[] list)	Returns a flat (rank-one) list comprising every item at every nest level within the given...
double Floor(replicatable double n)	Rounds a number down to the nearest whole number.
string Format(string formatString, ...)	Returns a string which is the result of applying the given format string to the given...
value GetPropertyValue(value qualifier, string...	Gets the value of a property.
bool InDynamics()	Returns true if the user is currently performing a dynamics operation.
value[] Intersection(value[] listA, value[] listB)	Returns a list of all items that are in both listA and listB.
void IsDebugging()	Returns true if we're currently stepping through code with the debugger.
value Last(replicatable value[] list)	Returns the last item in a given list.
value Limit(replicatable value n, value min,...	Returns a number limited to a given range.
value[] List(int count, optional value...	Generate a list whose items are populated with a value or by calling a function.
value[][] List2D(int count0, int count1,...	Generate a 2D list whose items are populated with a value or by calling a function.
value[][][] List3D(int count0, int count1, int...	Generate a 3D list whose items are populated with a value or by calling a function.
double Log(replicatable double n)	Returns the natural (base e) logarithm of a number.
double Log10(replicatable double n)	Returns the base-10 logarithm of a number.
value Max(replicatable value n1, replicatable...	Returns the larger of two numbers.
value Middle(replicatable value[] list)	Returns the middle item in a given list.
value Min(replicatable value n1, replicatable...	Returns the smaller of two numbers.
void OpenPrintFile(string filePath, optional...	Opens a text file. Subsequent Print and PrintFormat calls will write to that file.
double Pow(replicatable double x, replicatable...	Returns the result of x raised to the power y.
void Print(...)	Print the values of one or more expressions to the GCScript output pane.
void PrintFormat(string format, ...)	Print the values of one or more expressions to the GCScript output pane, using a format...
double RadToDeg(replicatable double radians)	Converts from radians to degrees.
value Random(optional replicatable int n1,...	Returns a random number.
int Rank(value[] list)	Returns the number of dimensions in a given value.
void RedrawAllViews(optional bool...	Redraws all of the MicroStation views.
value[] RemoveDuplicates(value[] list)	Returns a list of all unique items in the given list.
value[] ReverseSequence(value[] list)	Returns a list of all items in the given list, but in the opposite sequence.
double Round(replicatable double n, optional...	Rounds a value to a given number of places after the decimal point.
value[] Select(value[] list, bool function(value...	Returns a list of all items in given list that satisfy a given function.
GC.Feature[] SelectFeatures(int...	Returns a list of all features in the graph.
string Serialize(value any)	Returns the serialized representation of a value. The value can be recreated by calling...
value[] Series(replicatable value start,...	Creates a list from a given start number, final number and increment.
void SetPropertyFree(Feature feature, string...	Sets whether the specified property is free or unfree.
void SetPropertyValue(value qualifier, string...	Sets the value of a property.
void SetRandomSeed(int n)	Provide a starting seed for the random number generator.
void ShowMessageBox(string message)	Displays a message box and waits for the user the press the OK button.
int Sign(replicatable value n)	Returns -1, 0 or 1, based on whether a given number is negative, zero or positive,...

Abs(number) | Replace | Insert | Close

3.15 Snapshot of GC's Function base library

Name	Value	Analog Value
GC01_EXPDIV01	28	
GC01_EXPDIV02	100	
GC01_RADC	5	
GC01_TVALP01	0	
GC01_TVALP02	0.920	
GC01_TVALP03	0.850	
GC02_EXPDIV01	28	
GC02_EXPDIV02	100	
GC02_RADC	10	
GC02_TVALP01	0	
GC02_TVALP02	0.920	
GC02_TVALP03	0.820	
GC03_EXPDIV01	28	
GC03_EXPDIV02	100	
GC03_RADC	11	
GC03_TVALP01	0	
GC03_TVALP02	0.920	
GC03_TVALP03	0.830	
GC04_EXPDIV01	28	
GC04_EXPDIV02	100	
GC04_RADC	7	
GC04_TVALP01	0.980	
GC04_TVALP02	0.920	
GC04_TVALP03	0.900	
GC05_EXPDIV01	28	
GC05_EXPDIV02	100	
GC05_RADC	5	
GC05_TVALP01	0.970	
GC05_TVALP02	0.920	
GC05_TVALP03	0.880	
UTCS_H01	0	
UTCS_H02	15	
UTCS_H03	50	
UTCS_H04	65	
UTCS_H05	75	

3.16 Visual sliders aid in the manipulation of Global Variables and create a customizable user friendly modeling environment

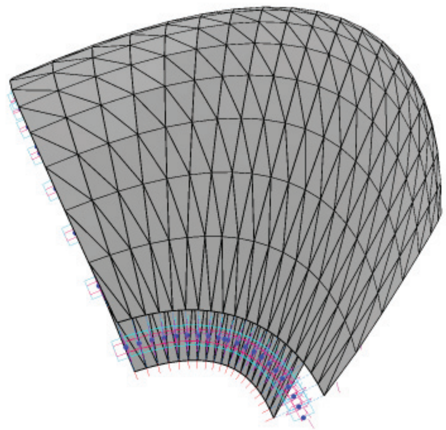


3.17 Variable Connection Diagrams

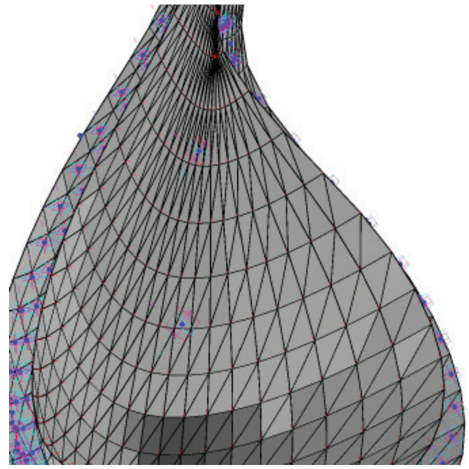
Base Components

The five primary *Components* of this exposition are exterior faceted cast-glazing (see 3.18), interior faceted alabaster (see 3.19), interior/exterior triangular truss systems (see 3.21), simple singular struts (see 3.20) and ball joints (see 3.22). The truss systems and struts are used to establish the space frame structure. The difficulty in modeling these *Components* was assigning the appropriate objective functionality. Learning and developing aspects of a tool through a practical exploration of geometry is an effective way of accelerating the modeling process. As the geometry is modeled it becomes refined over several iterations. For example, the simple triangular exterior and interior truss *Component* was initially designed by nine points, but through several iterations it became obvious that in order to provide added functionality and allow the model to become more explicit it required eighteen.

Each sub-section in this text explores the functionality of the five base *Components*. These *Components* become *Features* which are then applied to the base *Component system*.

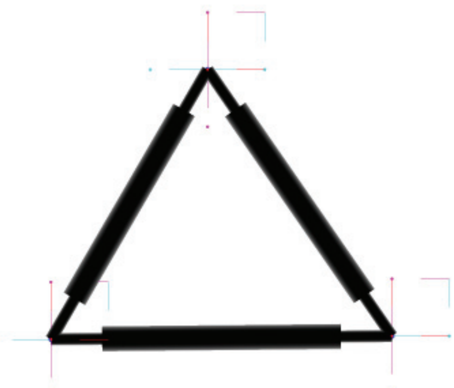


3.18 Exterior Facets (Triangular)

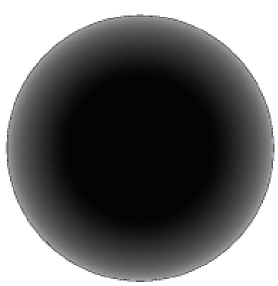


3.19 Interior Facets (Triangular)

3.20 Singular Strut



3.21 Triangular Truss



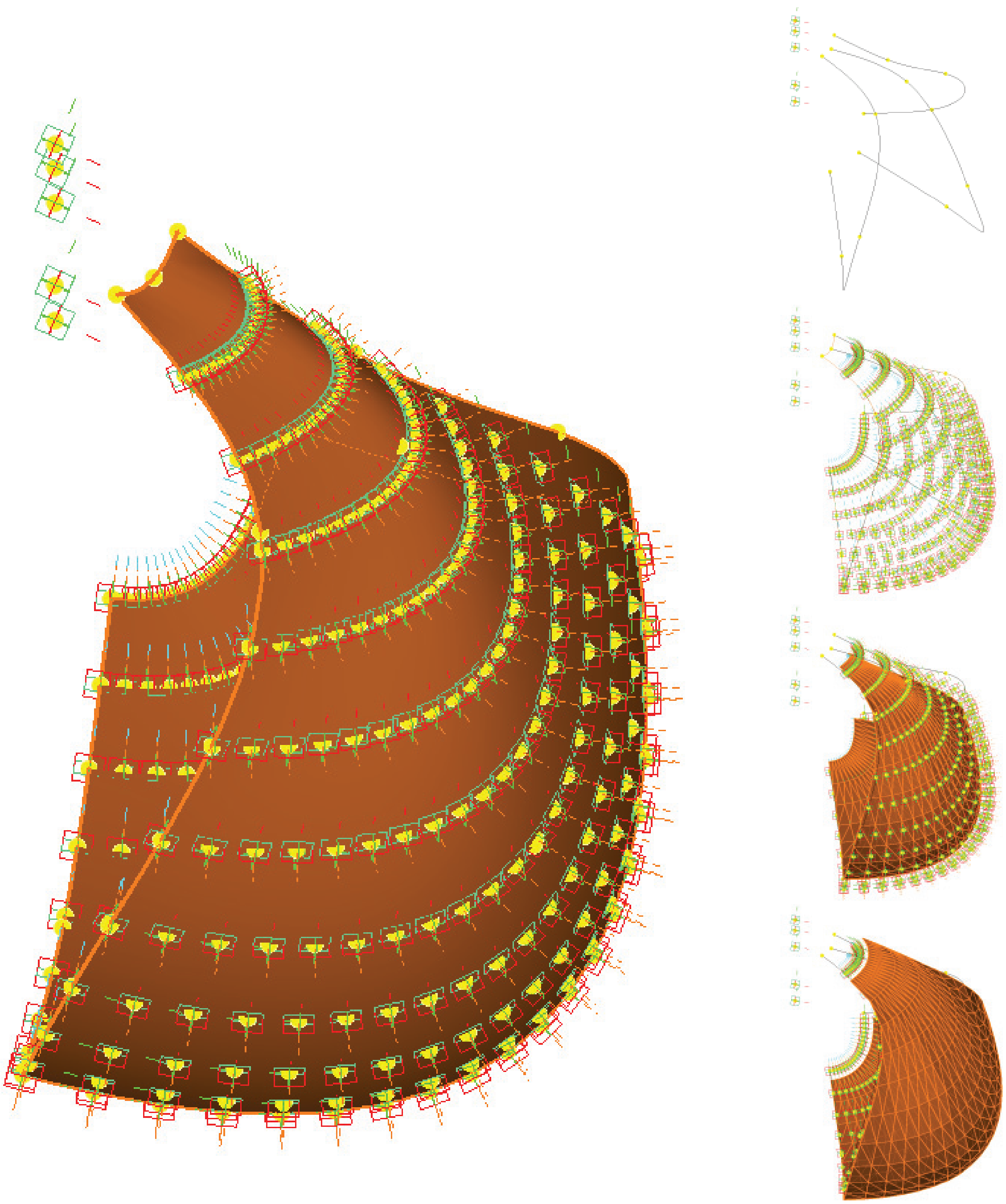
3.22 Ball Joint

Base System and Faceted Components

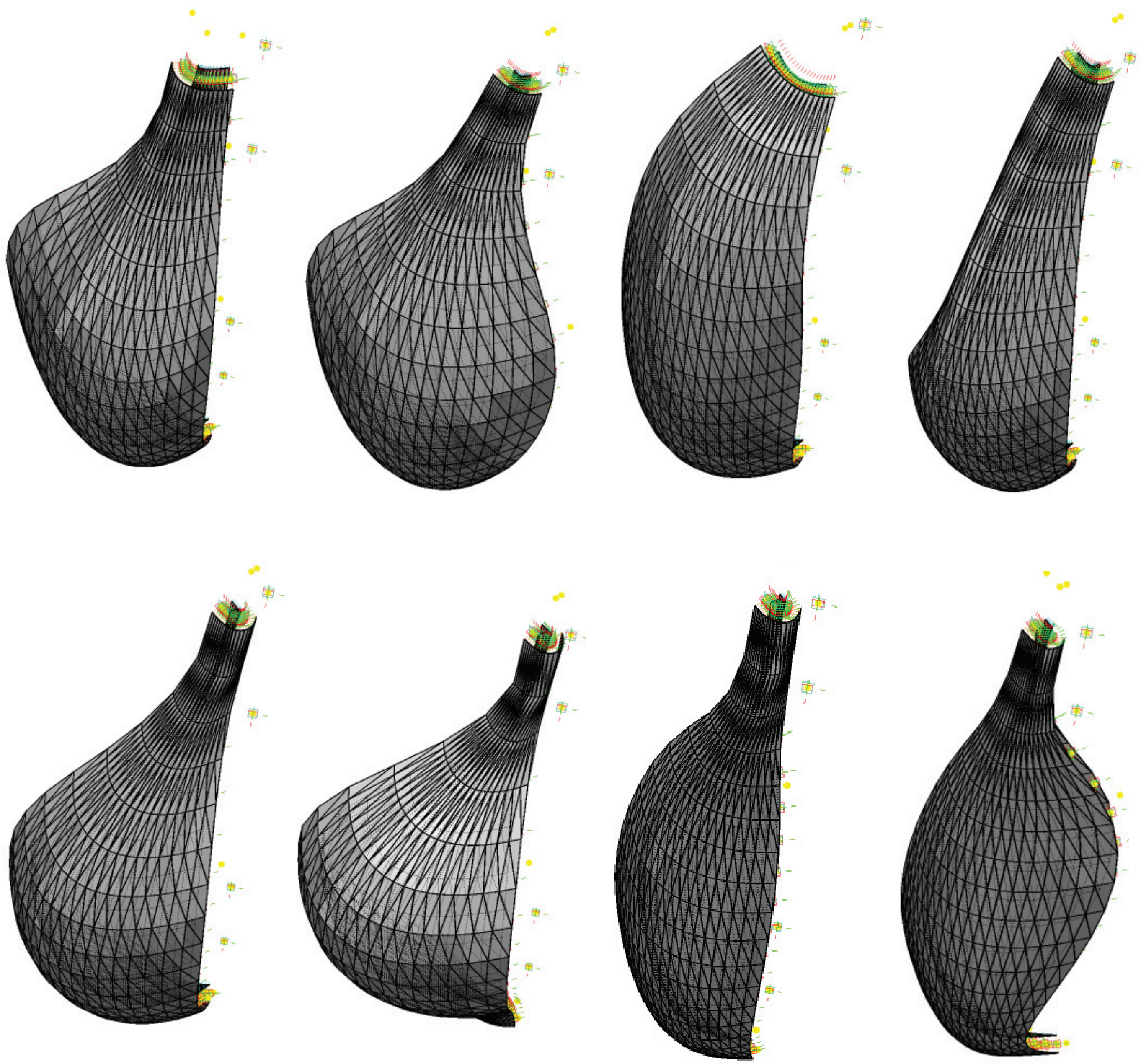
The interior and exterior facets are propagated amongst an array of points. The system that drives the endpoints is quite simple. The base component (as seen in 3.12) is arrayed vertically along the Z-axis. The coincidental points of these *Components* are used to create three unique *BSpline Curves*. The curves are then lofted to create the base *BSpline Surface*.

The *BSpline Surface* becomes the basis for the system that contains the remainder of the Fin. This surface can still be controlled by the base *Component* which allows the user to change the location of the points that drive the *BSpline Curves*. Changing these points effects both the surface and attached *Components*. Several coordinate systems (CS's) are propagated along the surface. There are 20 across the U and V coordinates. These CS's are used to determine the offset of the interior and exterior cladding from the *internal structure*. The coordinate systems are all *normal* to the adjacent surface, which clarifies the process by making sure that everything that is modeled on each coordinate system is perpendicular to the curvature of the surface. This in turn, ensures a direct offset.

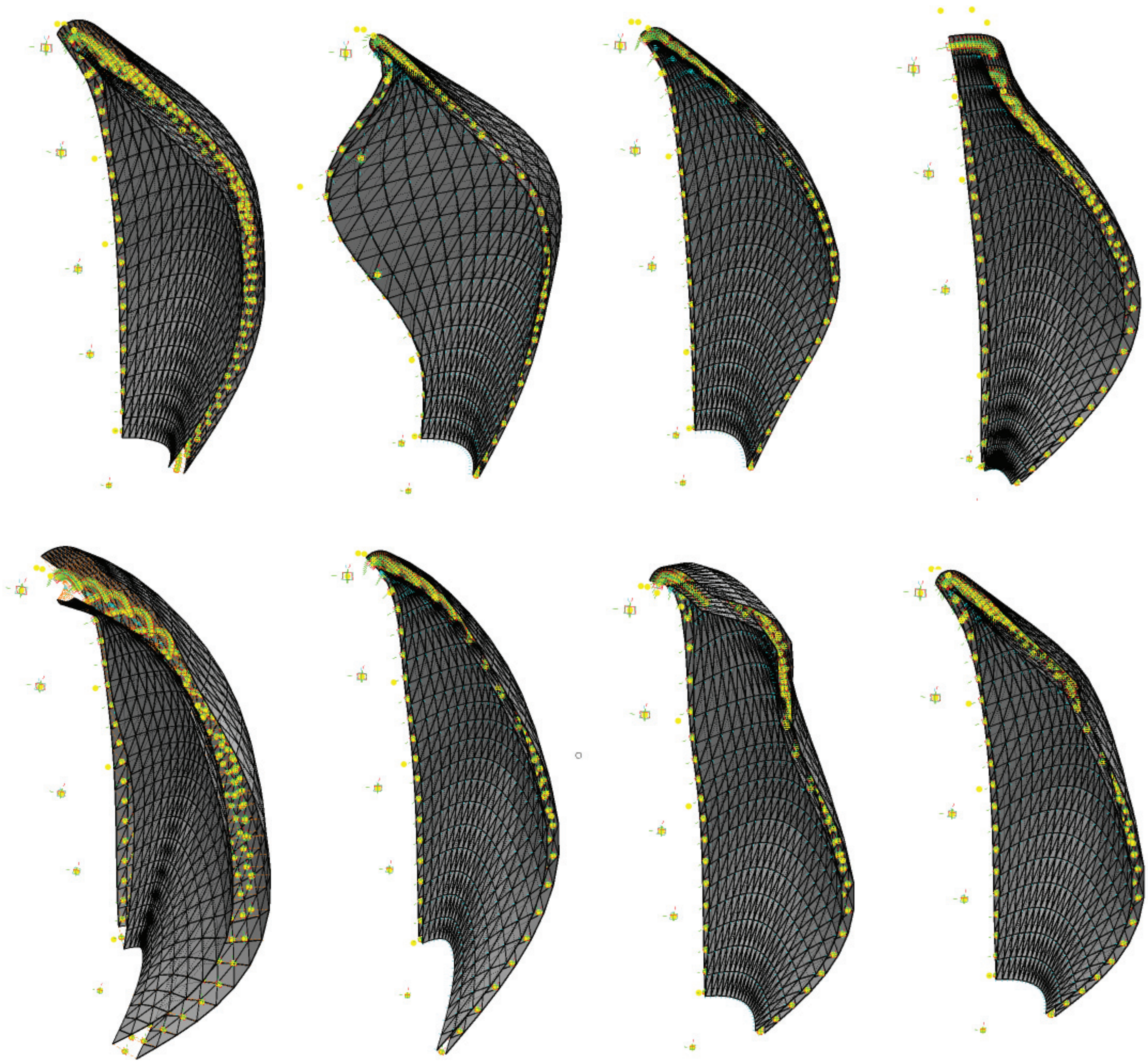
Once the coordinate systems and base surface are modeled we can prepare the interior and exterior faceted surfaces (alabster/cast-glass). To model a surface or shape points that define its boundaries must be selected. The coordinate systems could be used as the points of definition, but doing so would not address the requirement for a *surface offset variable*. To fulfill this requirement an additional component is required. A line which follows the *Z-vector* of each coordinate system completes the *Component*. Adding a *Graph Variable* to control the length of the line provides the ability to increase and decrease distance of the facets from the internal structural elements. This simple system is easy to manipulate and can be modified to include additional detailing.



3.23 CLOCKWISE FROM LEFT: Base Shape System and Surface; BSpline Curves Connecting Base Components; Coordinate Systems Populated on Base Surface; Interior Shape Facets with Offset; Exterior Shape Facets with Offset



3.24 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the exterior of the Fins



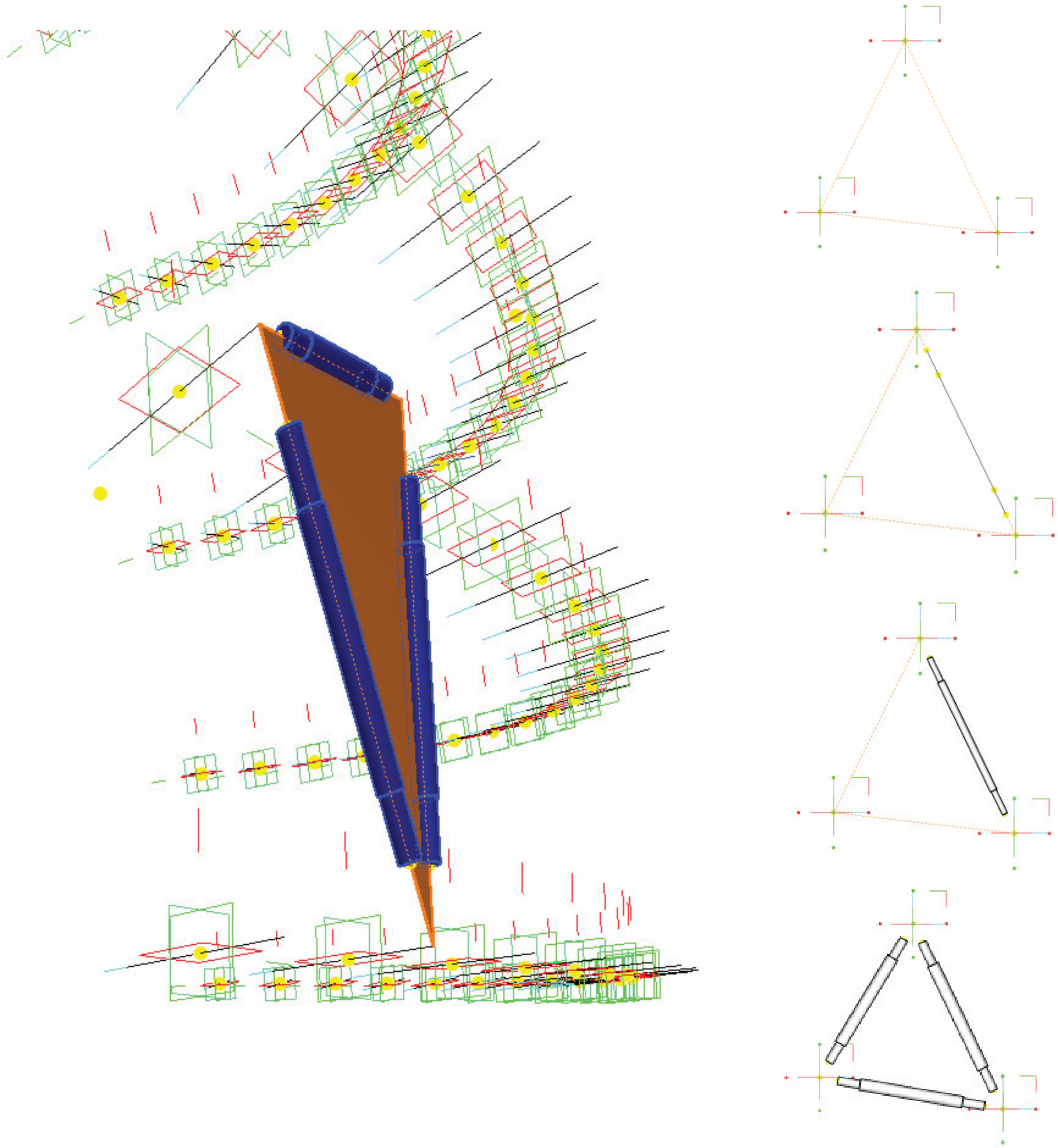
3.25 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the interior of the Fins

Triangular Truss Component

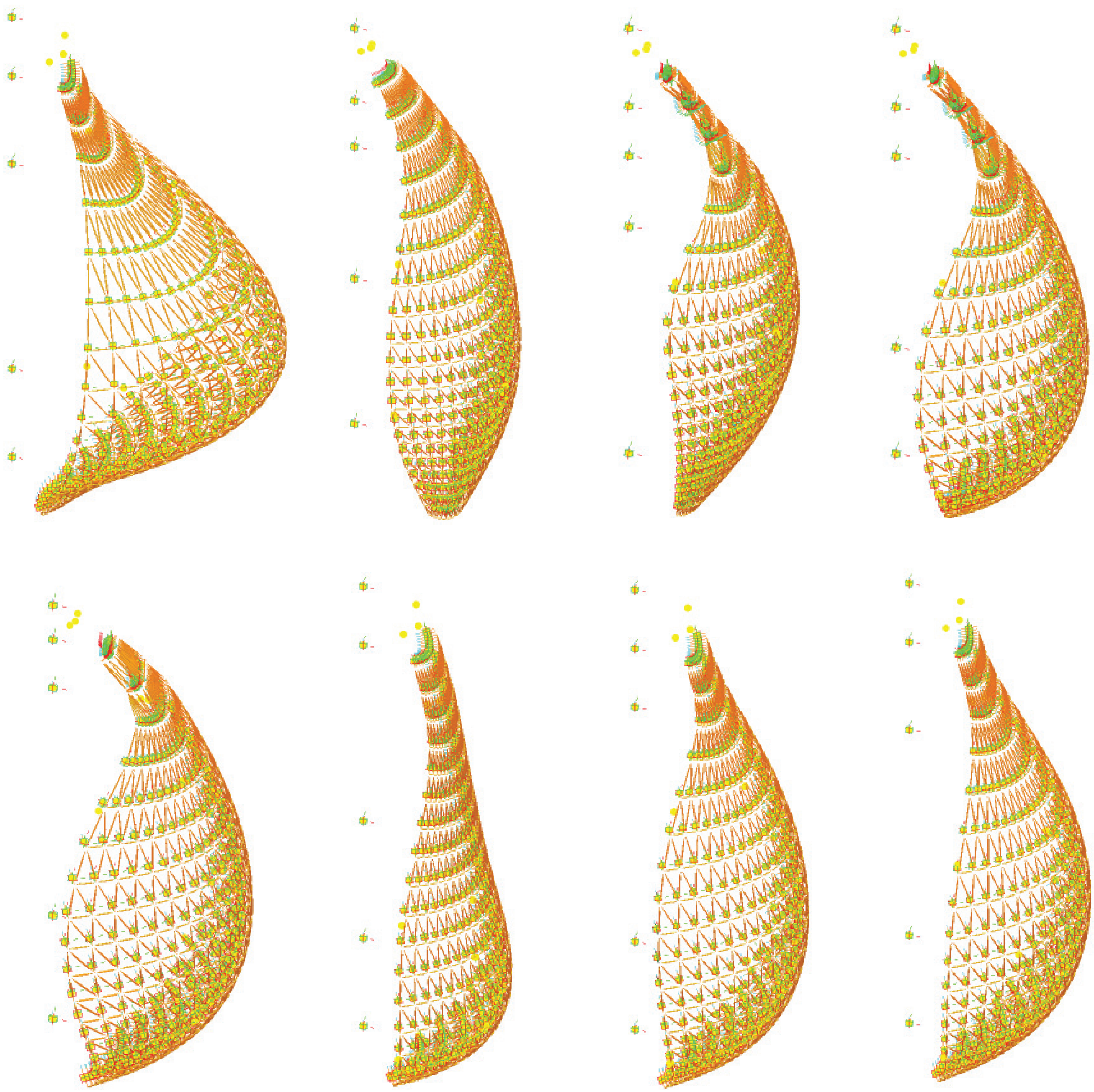
The triangular truss component is composed of a single element—a triangular shape. The shape's vertices act as endpoints for the three lines (one on each side) which act as rails for the four points that are bound to each of the aforementioned lines. The sectional radius of the middle and end members is proportional to their lengths. Start and End variables control the distance from the points of coincidence and a modifier is used to lock the length of the end struts. Changing these variables modifies the proportions of the truss components. This system is then remodeled to construct two more struts for the remaining sides of the triangle. A *Generative Component* (or *Feature*) could have been created from one of the struts of the triangular truss. It would have been possible to nest this *Component* within the triangular truss *Component*, alleviating the need to model three struts within the same *Transaction Script*. Nesting GC's makes it incredibly difficult to access *Graph Variables* once the GC is recompiled. Once the triangular truss model is complete it can then be compiled as a *Generative Component*.

The singular truss component is simply a *Feature* consisting of one truss member. Slight modification of the member's radius equation is required to maintain proportional member sizing.

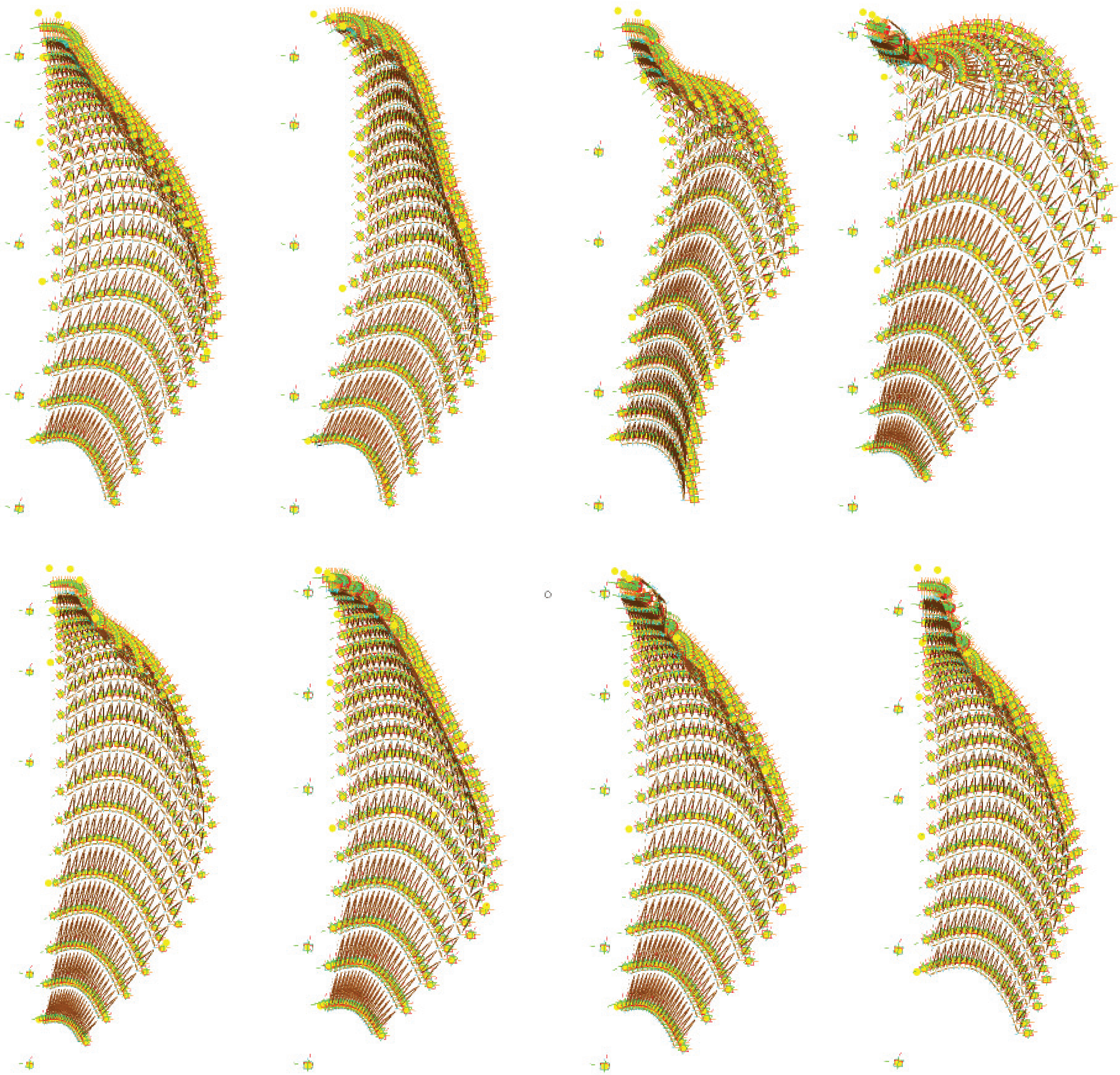
The truss component incorporates the same base system that is used to propagate the interior and exterior cladding model. In fact, the system used for the cladding model is almost identical. The only difference is that the line length used to offset the surfaces is slightly modified in the triangular model. The shape used to propagate the truss model is a triangular shape array. The basis of the shape array is modeled on the array of end points attached to the coordinate systems. This allows the truss component to be populated across the surface, which creates an array of structure that can be controlled or modified from three implicit variables—the results of these scripts produce the space frame structure.



3.26 CLOCKWISE FROM LEFT: Single Triangular Truss Component; Base Shape and Control Points; Base Lines and Support Lengths; Single Strut; Triangular Truss Component.



3.27 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the exterior triangular truss system of Components

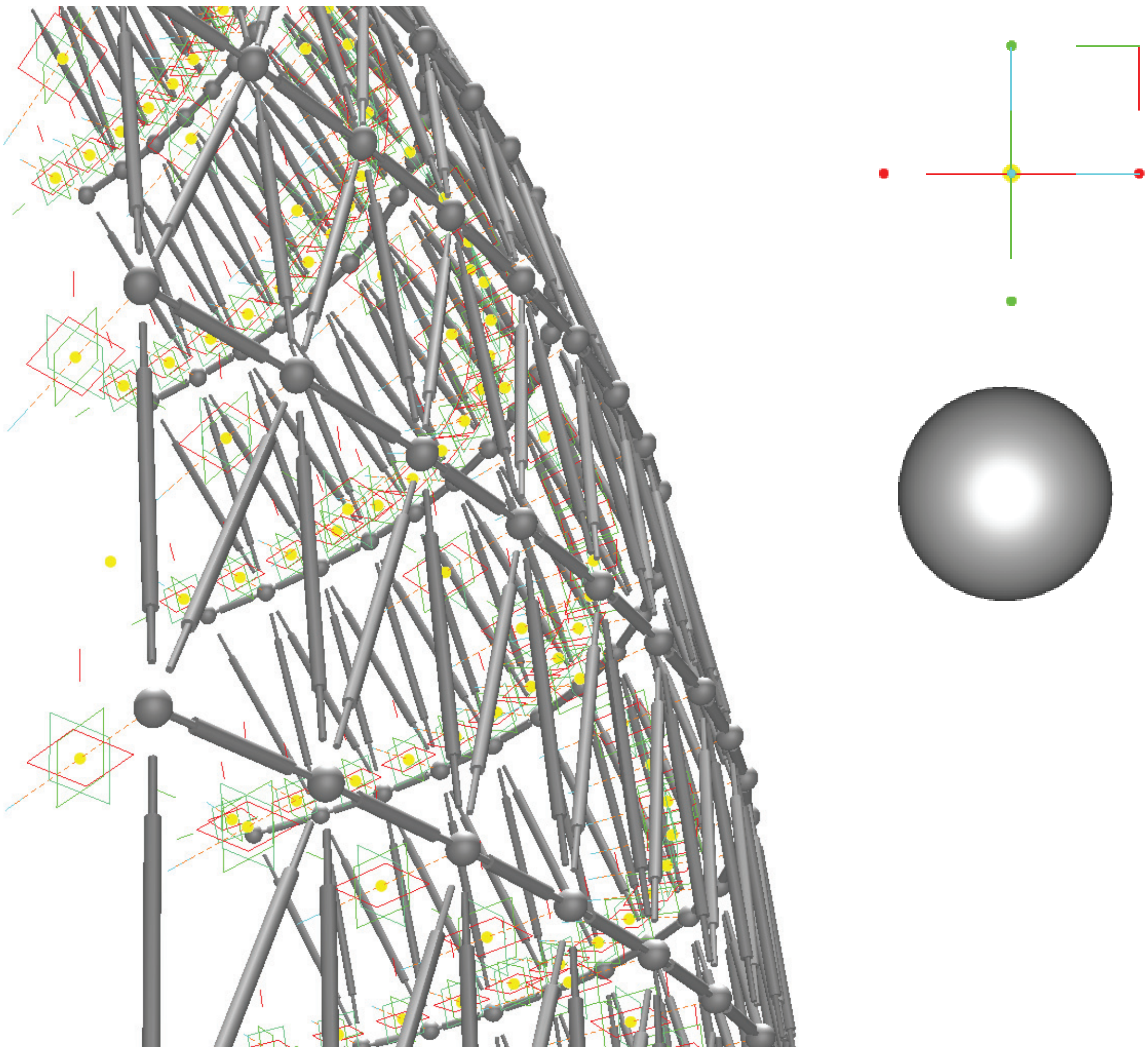


3.28 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the interior triangular truss system of Components

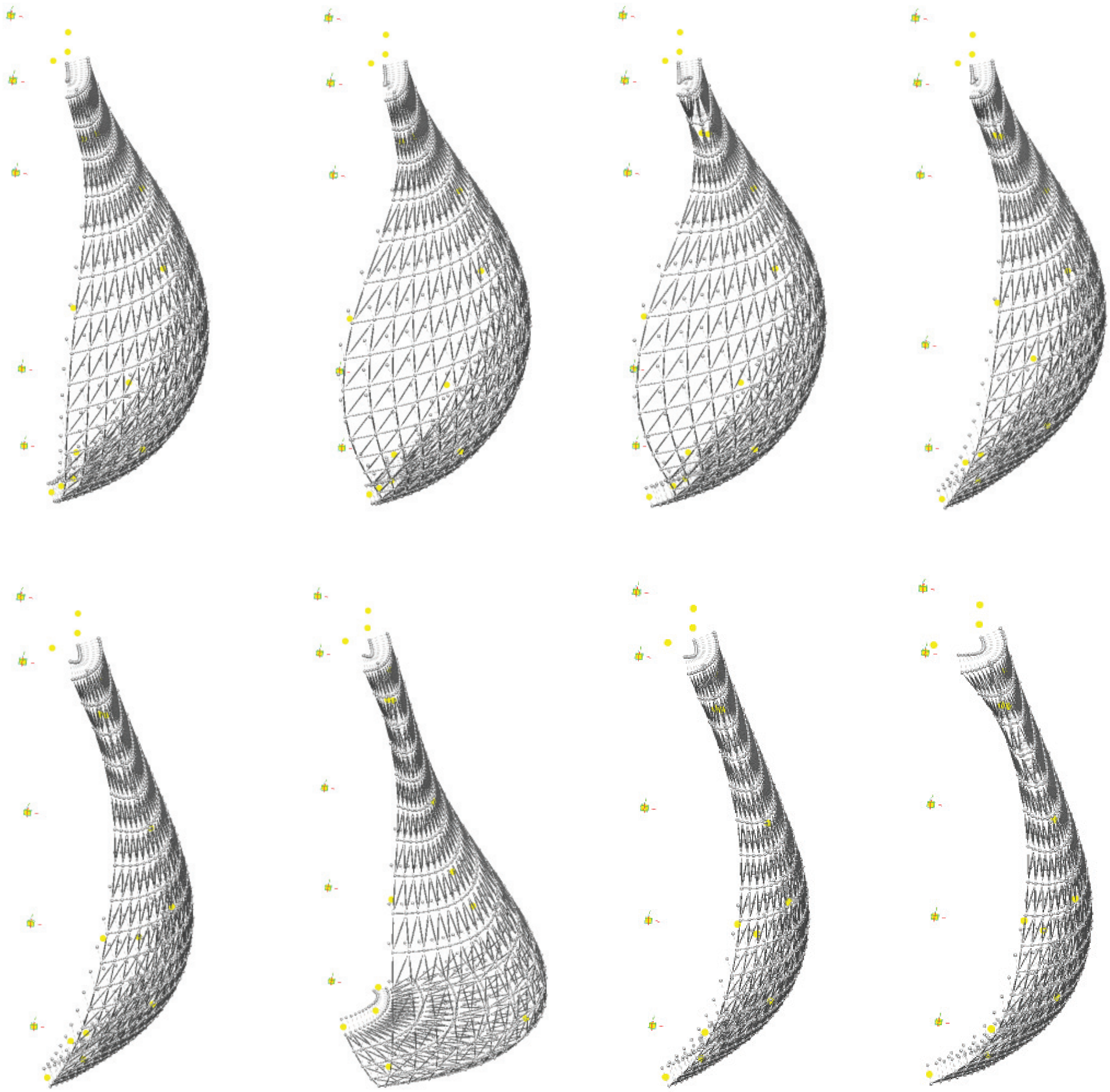
Ball Joint Component

The ball joint *Component* is simply a solid sphere modeled atop a base point. The ball joint *Feature* has only one variable—scale. Very little detail is required for this *Component*; its size is what influences the truss spacing and this determines the size of the truss members.

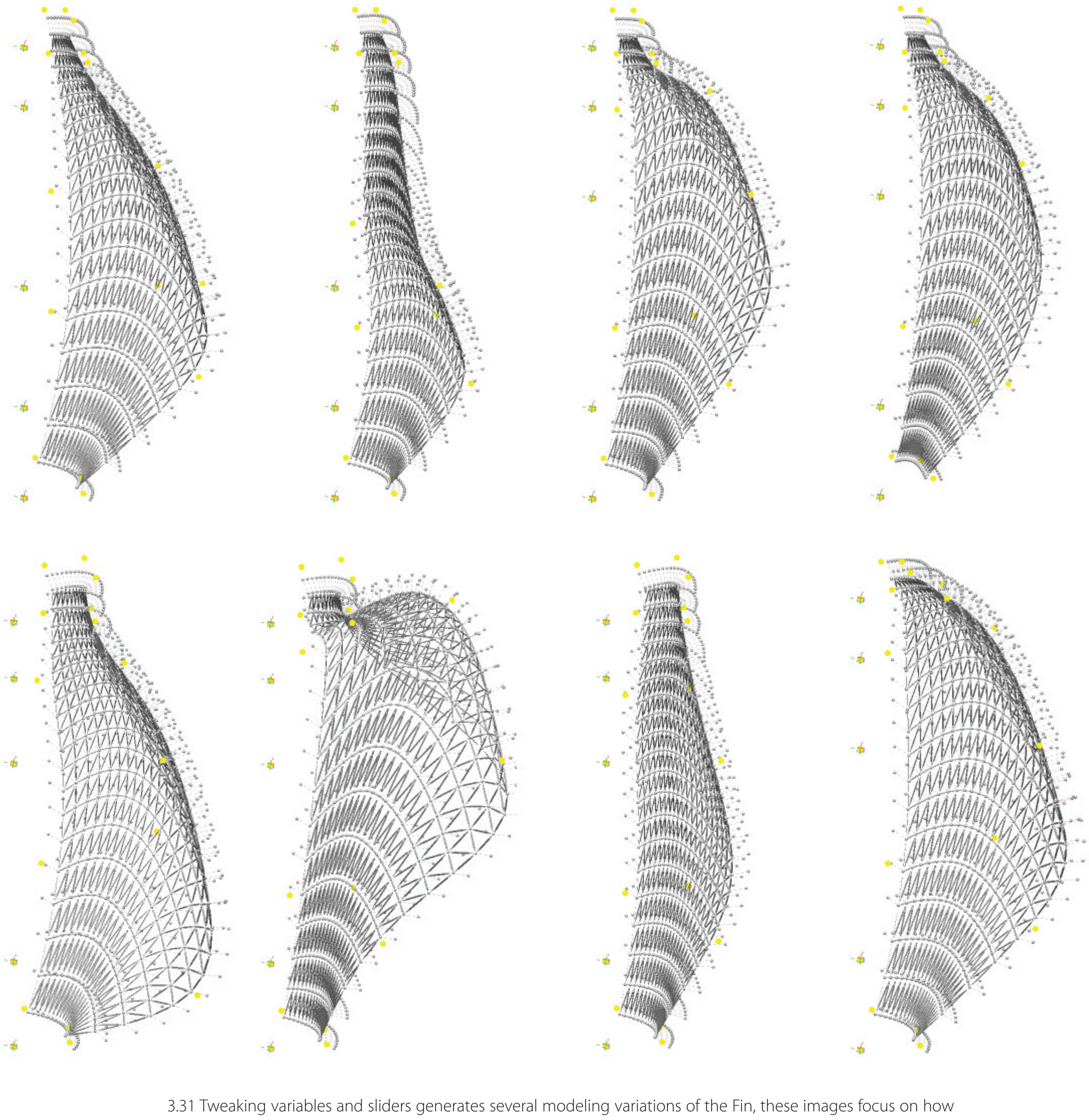
The ball joint is propagated amongst the structure at each endpoint of the interior and exterior offset lines. This ensures that each joint has members that are normal to where they connect.



3.29 CLOCKWISE FROM LEFT: Propagated Joint Component; Ball Joint Base Point; Ball Joint



3.30 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the exterior composition of structural Components, including the triangular truss system and ball joint



3.31 Tweaking variables and sliders generates several modeling variations of the Fin, these images focus on how this variation has effected the interior composition of structural Components, including the triangular truss system and ball joint

COMPLEX ARCHITECTURES

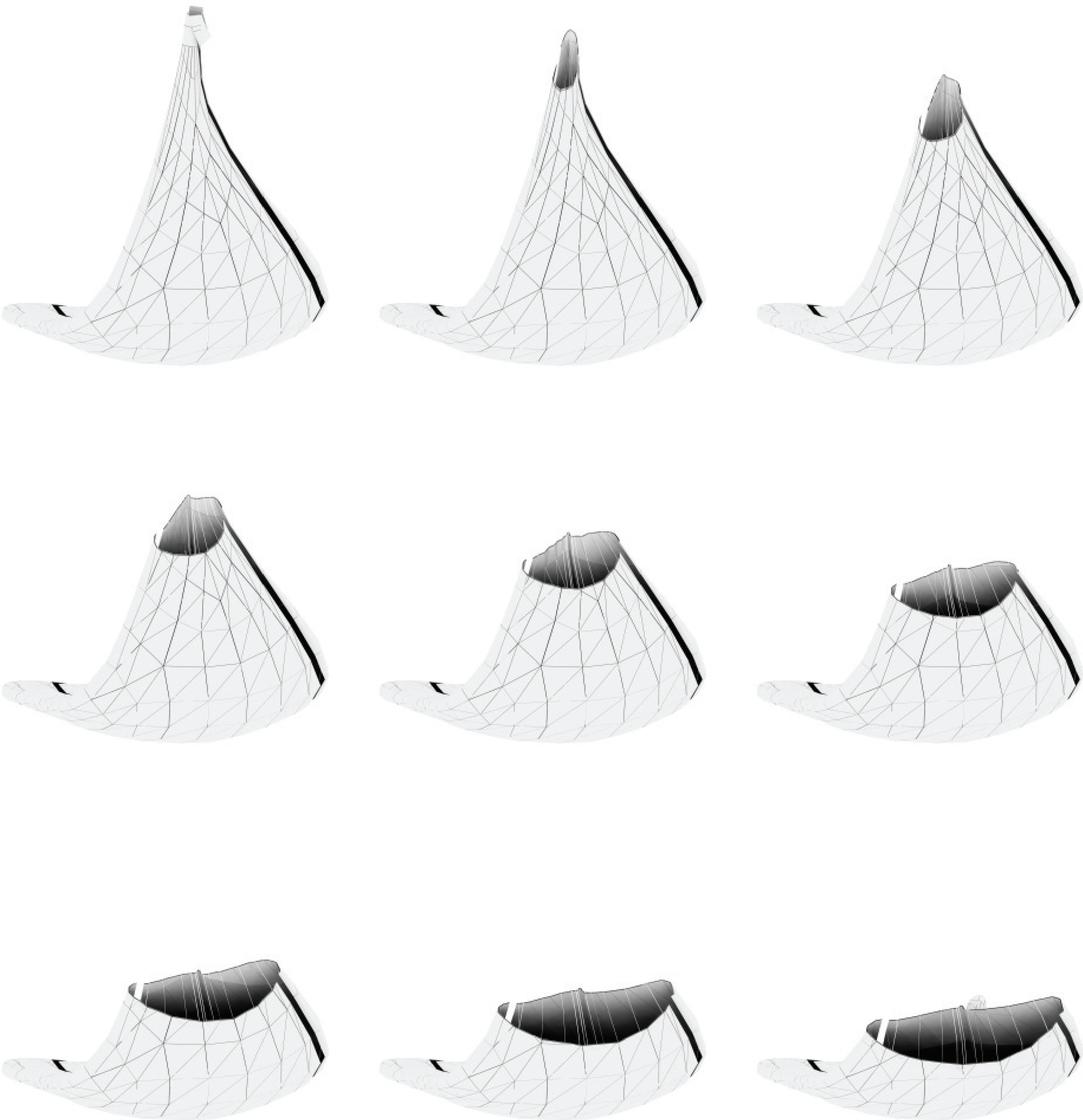
NEXT *3.32 The following sections reveal the change in composition of a Temple Fin, they are cut at one metre intervals (from CATIA).*

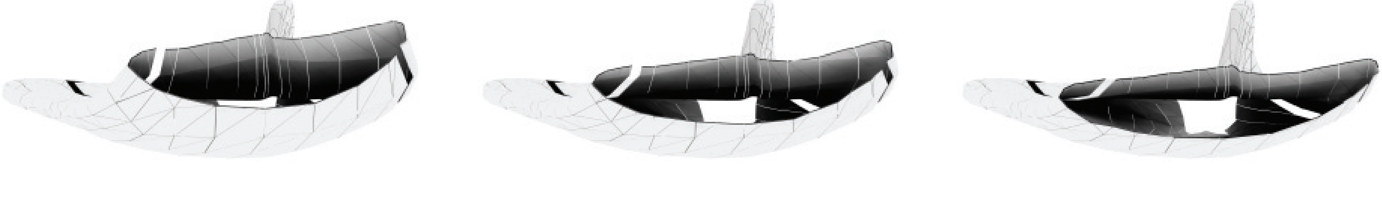
This segment of the chapter builds upon the previous content by nesting *Components* within one another. This facilitates the extraction of data which can be used to fabricate components, plot the location of key points in space or give a general numerical assessment of the model. Alternative control methods are explored as a means to adjust variables via different methods. The most prevalent method is via a *Law Curve*. This segment expands on key features available in Generative Components and explores alternative control and rigging methods.

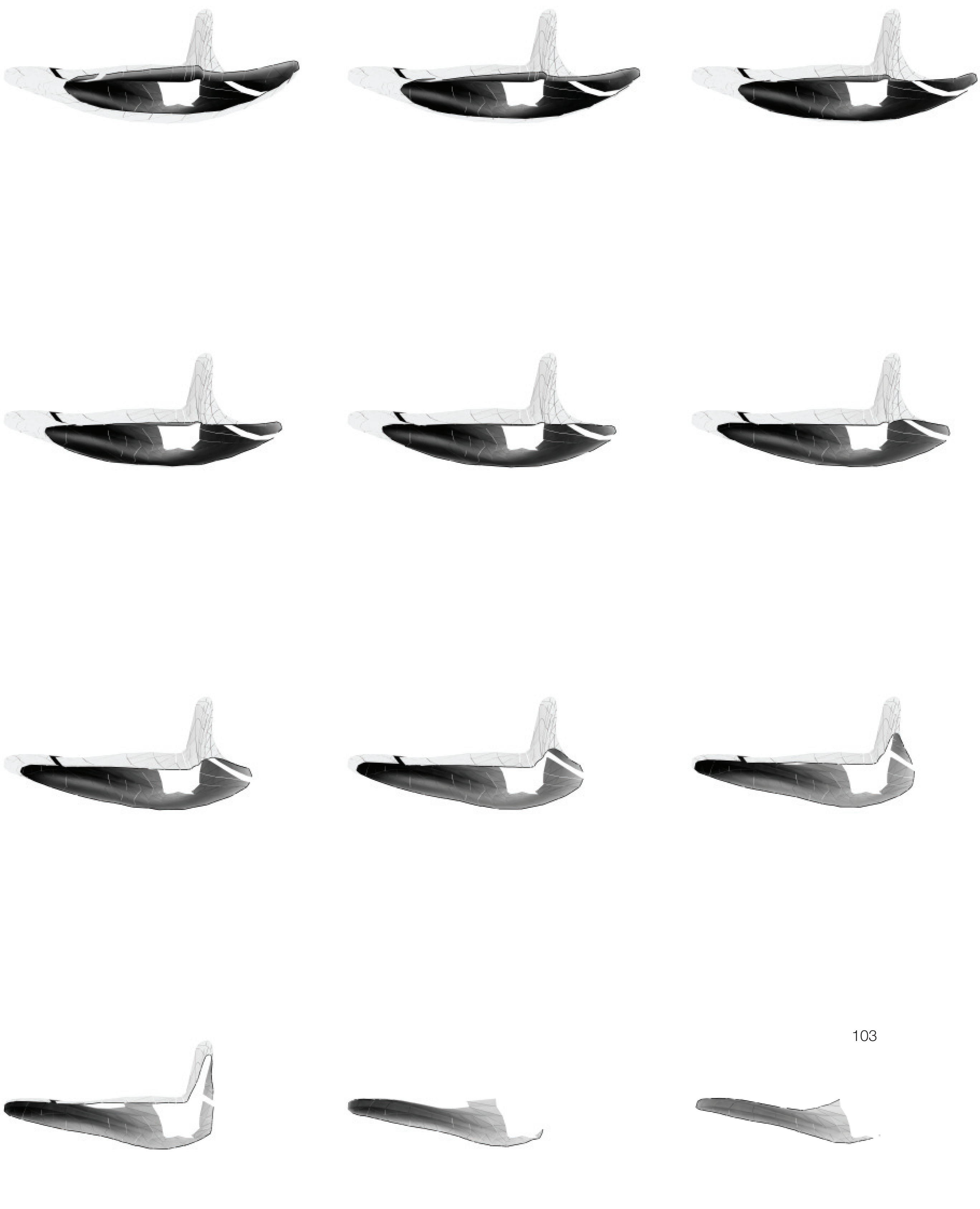
Complex Architectures are objects that contain several levels of depth and variation, this definition can be applied to both *geometric* and *application models*. Designing complex models produces adaptable geometry. It increases the amount of designer intervention, which enhances *Components* and the overall architecture of the model.

Generative Components has many tools that encourage the input and output of data in several different ways. Numerical or geometric information that is read or written by Generative Components can be inputted or outputted in the form of Excel spreadsheets. GC allows you to create rich and *robust* variable driven geometry (as shown in previous figures) with relative ease.

The previous geometric models (see 3.23, 3.26 and 3.29) were designed to be *robust* through the integration of *Functions* and *Graph Variables*. Designing the system in this manner allows the designer to use existing software functionality as well as custom functionality to enhance both *top-level components* and *sub-components*. The following figures build upon the base geometric *Components*, increasing the level of geometric refinement and data extraction.



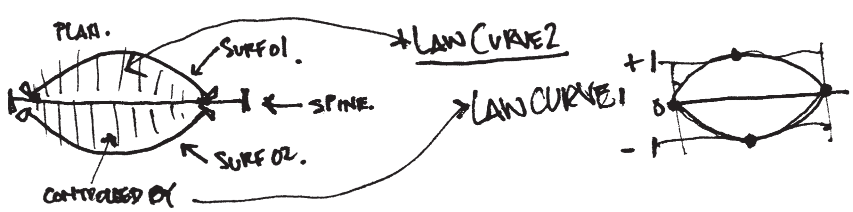




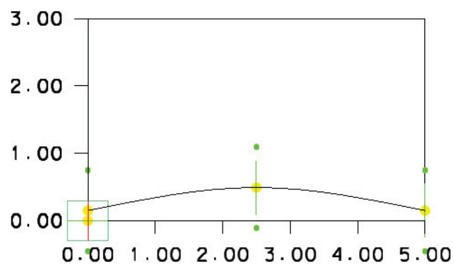
Details, Complexity and Information Extraction

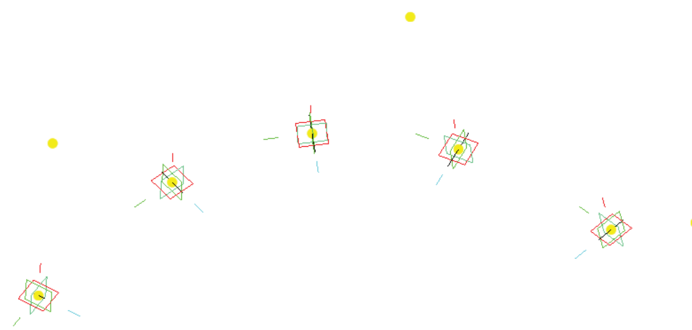
The interior and exterior CATIA geometry of the Bahá'í Temple is curved on all three axes. The surfaces taper as they near edge boundaries creating a unique section (see 3.32). Suppose we want to control the section; in Generative Components the most graphical means of doing this is through the use of a *Law Curve*. *Law Curves* allow the designer to control almost any aspect of a *Feature* through the use of *Dependant* and *Independent* variables. The variables that are extracted from the *Law Curve* are extrapolated based on the curve's position inside its frame (see 3.33). The following figures (see 3.33-3.38) use the profile of a *Law Curve* to determine the shape of the surface and the distance the surface is offset from its point of registration.

The amount of complexity involved in the structural *Components* of the Temple Fin is remarkably large. Generative Components has the ability to read and write spreadsheet data. This data can be used to plot the location of integral design components in three dimensional space (see 3.44 and 3.46), which provides a numerical understanding of the architecture.

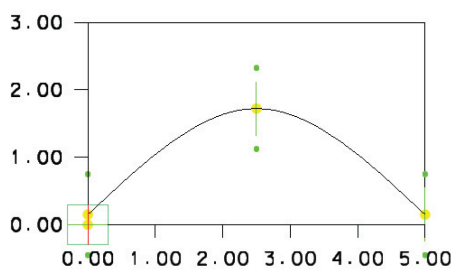


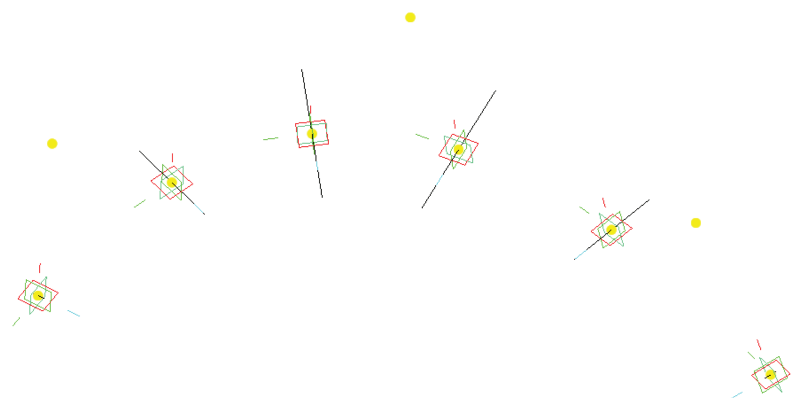
3.33 Preliminary sketch illustrating effects of Law Curve application—as the curve is adjusted the deflection of the inner and outer skins would increase or decrease



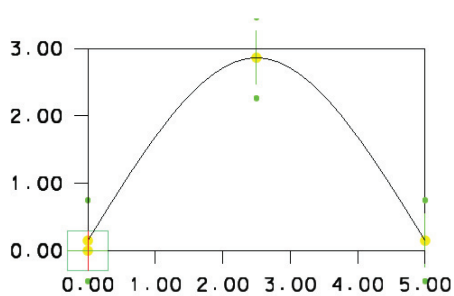


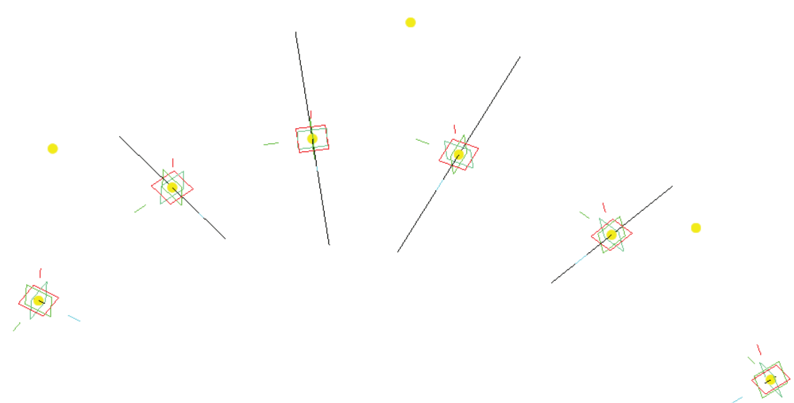
3.34 Law Curve Controlling Surface Offset Geometry
(0.5 units)—line lengths increase marginally



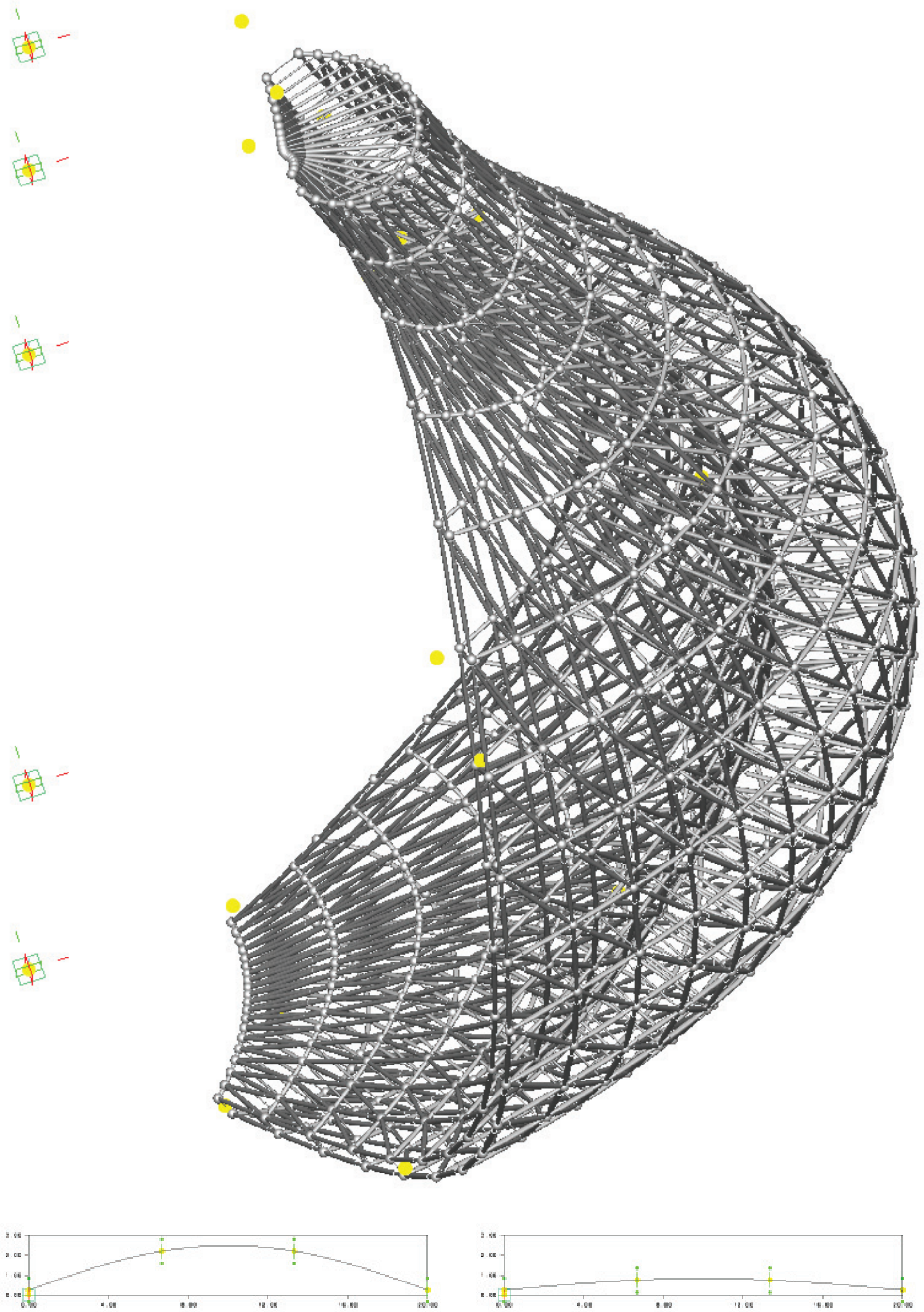


3.35 Law Curve Controlling Surface Offset Geometry
(1.5 units)—line lengths increase moderately

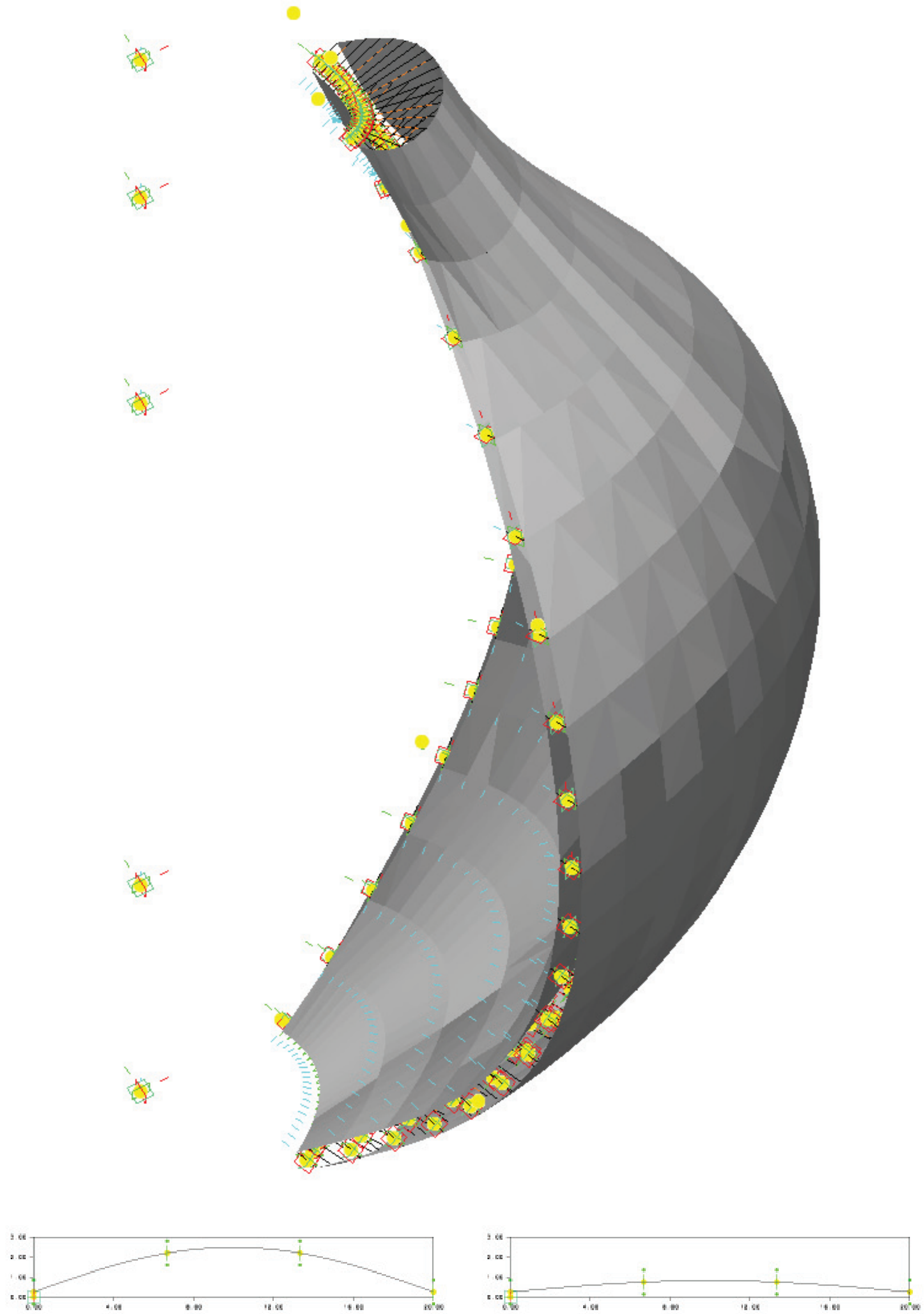




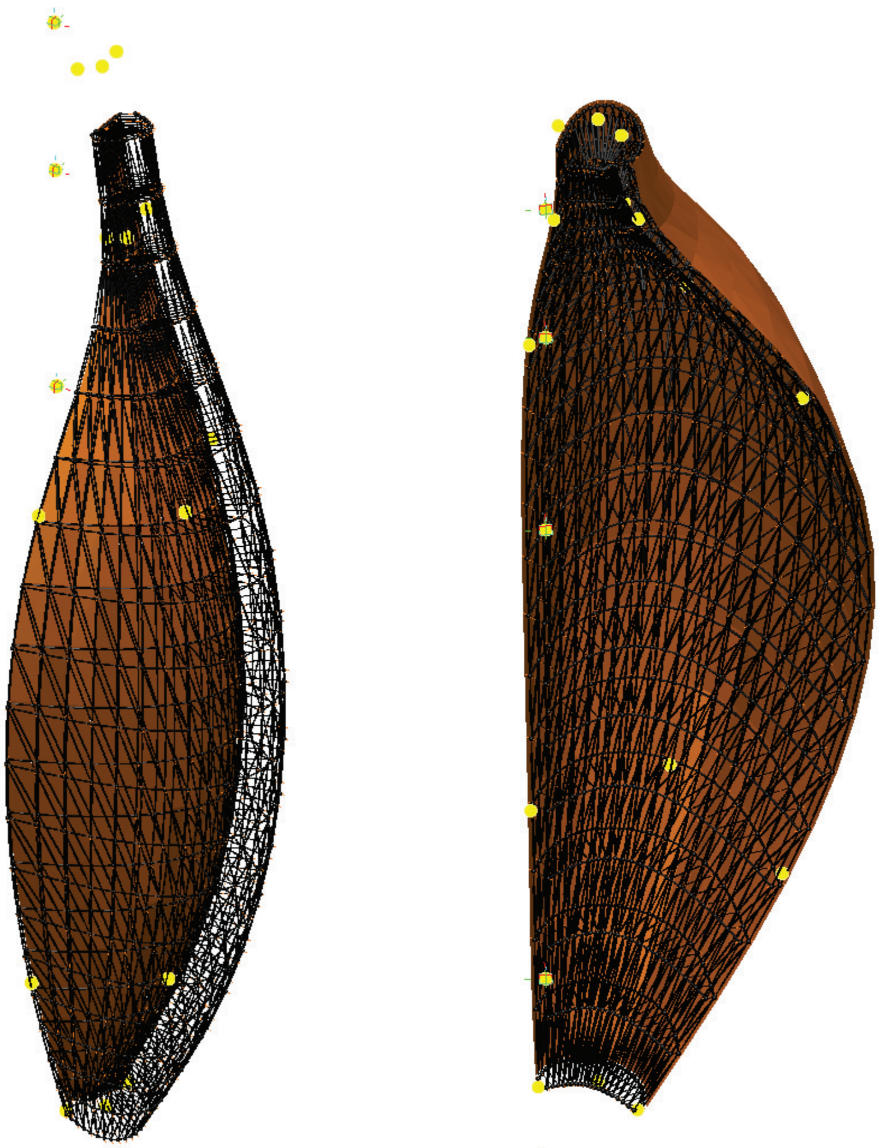
3.36 Law Curve Controlling Surface Offset Geometry
(2.8 units)—line lengths increase exponentially



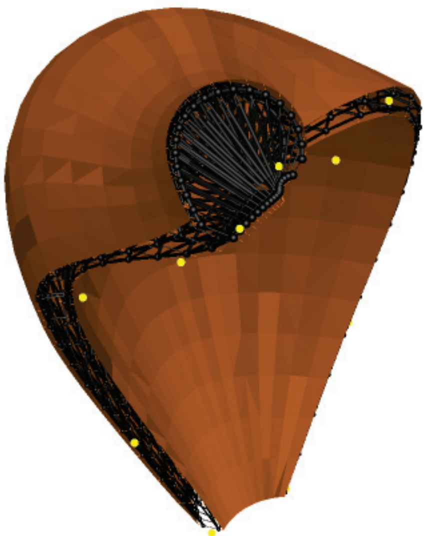
3.37 The effects of the Law Curve can be seen on the illustration above—compare the shape of the curve to the line of curvature on the interior and exterior surfaces, they are a visible match (diagonal support struts were removed to increase the visibility of the main structural components)



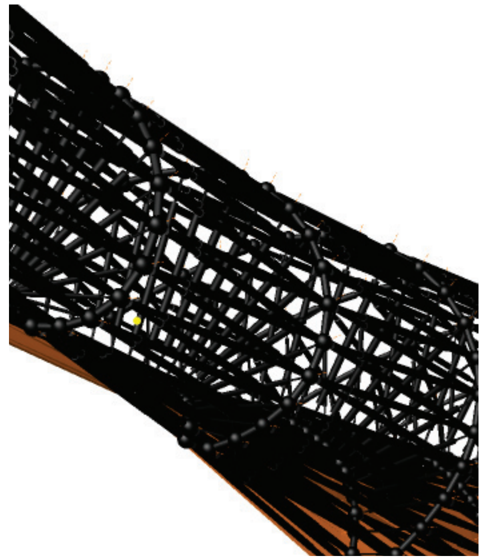
3.38 The effects of the Law Curve can be seen on the illustration above—compare the shape of the curve to the line of curvature on the interior and exterior surfaces. The results are more noticeable in the previous example due to the grid-like structure



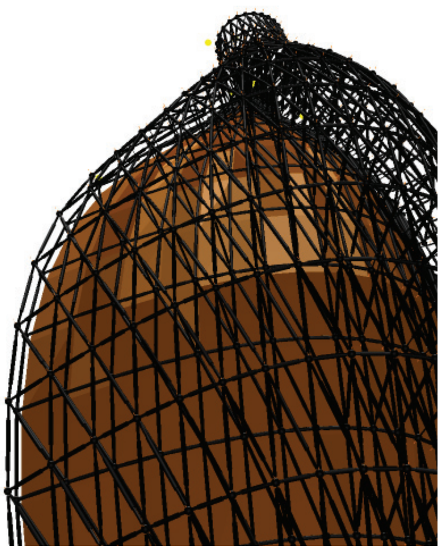
3.39 Interior and exterior illustrations represent the relationship of the inner and outer skins to the structure (diagonal support struts were removed to increase the visibility of the main structural components)



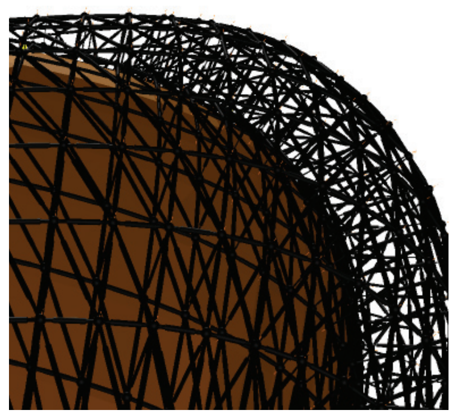
3.40 Complete Fin with additional structural elements



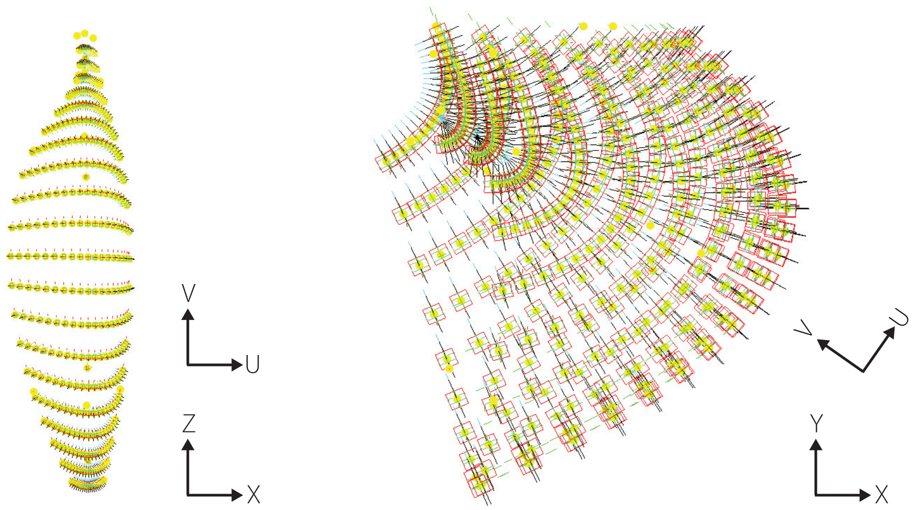
3.41 Fin neck detail



3.42 Exterior view of the structure



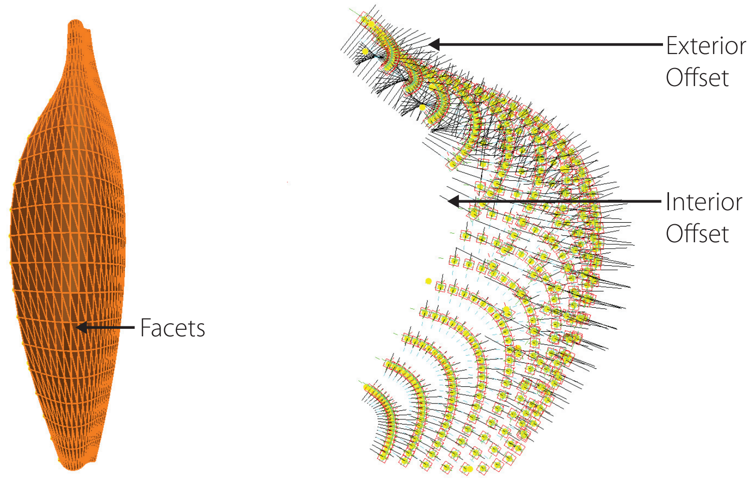
3.43 Exterior detail of the structure



3.44 Coordinate system (CS) location and UV parameter diagrams—the data in the table can be used to determine the location of the coordinate systems relative to 0,0,0; the table addresses a single row of CS's

CS X CO-ORD	CS Y CO-ORD	CS Z CO-ORD	CS U PARAM	CS V PARAM
3.326	-4.578	0	0	0
3.892	-6.358	3.449	0.05	0.05
4.435	-8.095	6.934	0.1	0.1
4.935	-9.746	10.491	0.15	0.15
5.368	-11.269	14.156	0.2	0.2
5.715	-12.621	17.959	0.25	0.25
5.982	-13.764	21.884	0.3	0.3
6.184	-14.661	25.895	0.35	0.35
6.337	-15.275	29.953	0.4	0.4
6.458	-15.569	34.022	0.45	0.45
6.563	-15.507	38.064	0.5	0.5
6.668	-15.051	42.040	0.55	0.55
6.788	-14.165	45.915	0.6	0.6
6.941	-12.813	49.649	0.65	0.65
7.126	-10.998	53.222	0.7	0.7
7.261	-8.947	56.690	0.75	0.75
7.235	-6.961	60.141	0.8	0.8
6.937	-5.341	63.659	0.85	0.85
6.267	-4.362	67.323	0.9	0.9
5.277	-3.970	71.126	0.95	0.95

3.45 Coordinate system variables written to spreadsheet from Generative Components



3.46 Facet area and offset diagrams—the following diagrams address the location surface facets and interior/exterior offsets; the offsets are the numerical value of the distance of the skins from central spine of the array

EXT FACET AREA	INT FACET AREA	EXT OFFSET	INT OFFSET
0.740	0.622	0.25	1.1
1.191	1.076	0.695	0.828
1.647	1.522	1.118	0.575
2.102	1.962	1.500	0.361
2.552	2.394	1.817	0.206
2.961	2.791	2.051	0.131
3.313	3.132	2.181	0.160
3.571	3.377	2.187	0.319
3.743	3.538	2.072	0.610
3.791	3.571	1.868	0.996
3.733	3.505	1.610	1.429
3.547	3.304	1.333	1.861
3.240	2.995	1.071	2.244
2.803	2.579	0.856	2.533
2.199	2.042	0.718	2.687
1.553	1.423	0.660	2.710
0.947	0.812	0.670	2.629
0.556	0.331	0.731	2.465
0.455	0.057	0.833	2.243
		0.960	1.981
		1.1	1.7

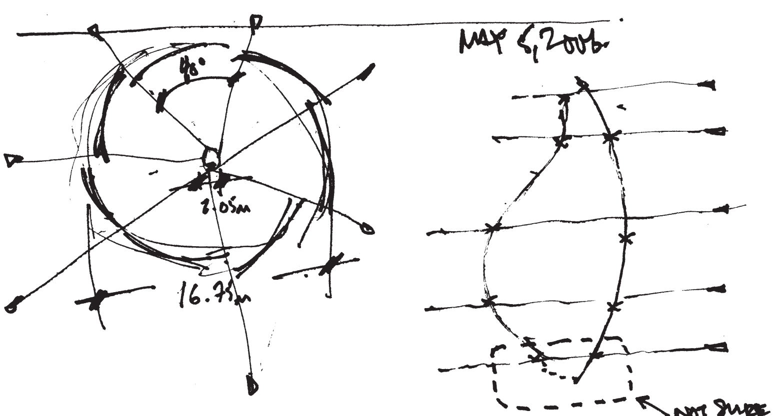
3.47 Coordinate system variables written to a spreadsheet from Generative Components

Continuous Process

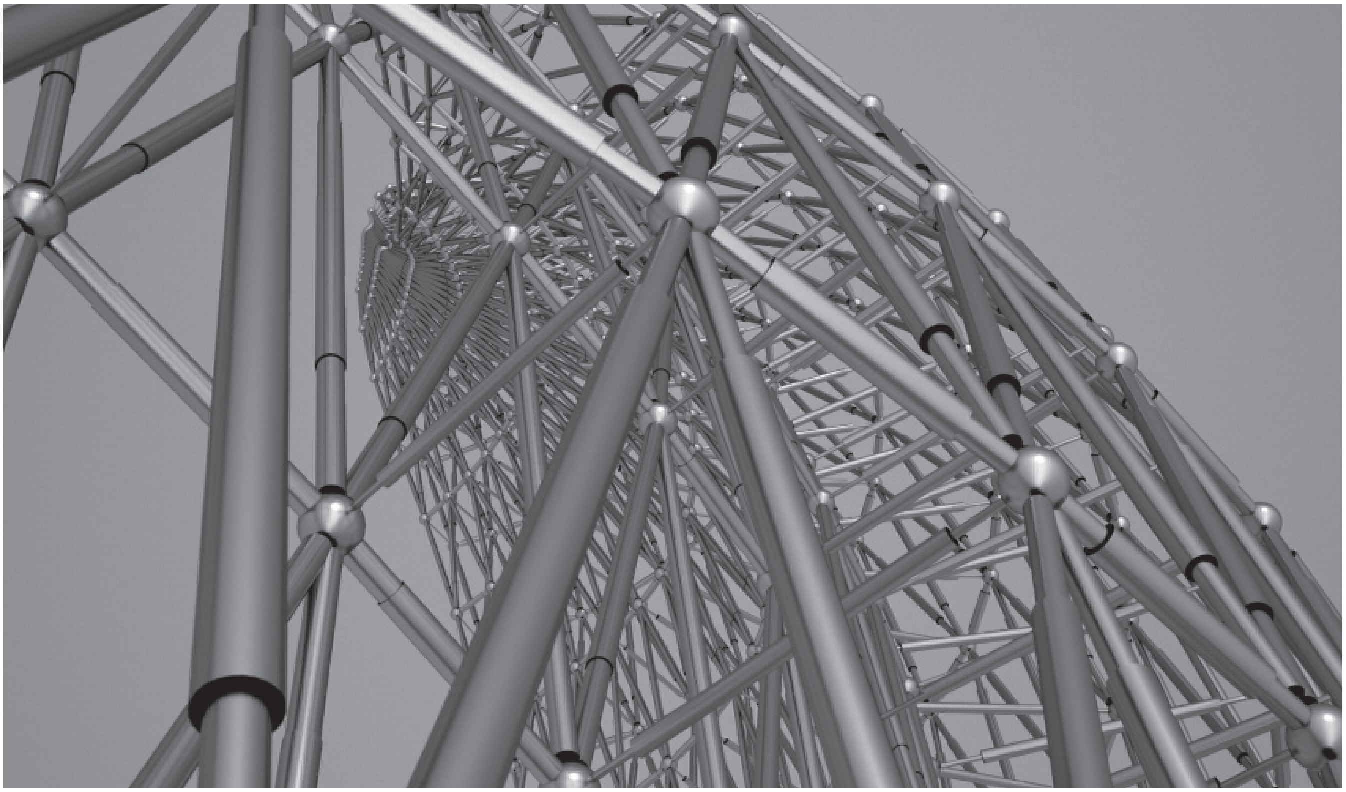
These Components are still works in progress. Generative Components clearly attempts to position itself as a tool built for parametric designers. The program itself is a work in progress..

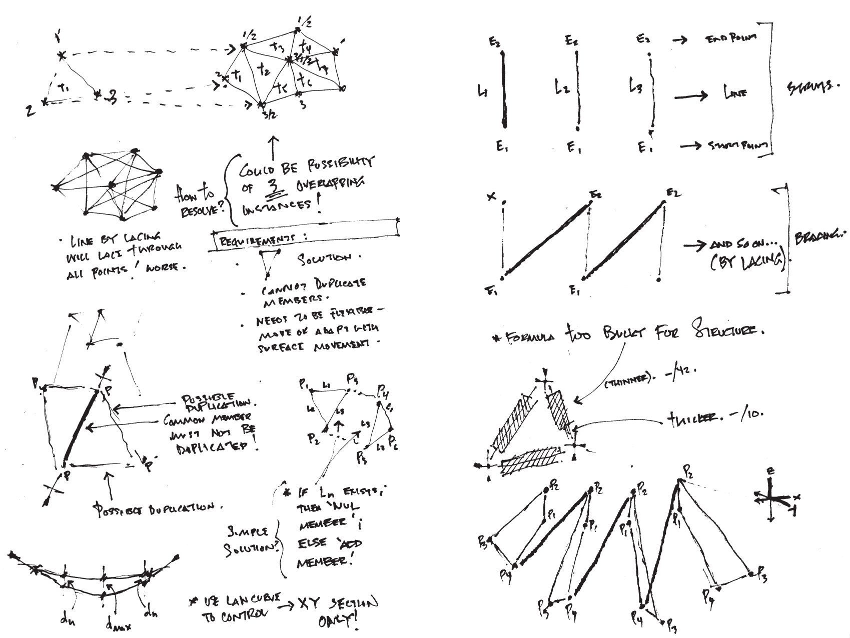
The term "work in progress" implies that something is a fragment. The Temple is composed of hundreds of fragments, some of which—even at its current advanced design stage—have yet to be considered. The accompanying sketches (3.48-3.51) illustrate the conceptual stages of organization when creating a parametric design model.

RIGHT (CLOCKWISE FROM TOP) *3.49 Detail of GC structure (angle struts removed for clarity), 3.50 Conceptual strut diagrams representing the development of the triangular truss Component, 3.51 Truss resolution sketches documenting member sizing formulas and ratios used in GC.*



3.48 Conceptual Fin sketches, used as a GC planning exercise





How to resolve?
Could be possibility of 3 overlapping instances!
· Line by lacing will lace through all points! worse.
Requirements:
Solution.
· Cannot duplicate members.
· Needs to be flexible – move or adapt with surface movement.
Possible duplication.
Common member must not be duplicated!
Possible duplication.
* If Ln exists; then 'nul member'; else 'add member'!
Simple solution?
* Use lancurve to control → XY section only!
End point
Line
Start point
And so on... (by lacing)
Bracing.
* Formula too bulky for structure.
(thinner). –/42.
thicker. –/10.

A Summary of the Process

This exposition tests both fundamental and advanced features of Generative Components. It challenges the response time of the tool through the application of extremely dense data-sets—the Fin structure contains over ten thousand components. The exposition implements unique *Features* and scripts which provide a visual likeness to the CATIA model while establishing unique model making principles. The *Features* and scripts are used to enhance the model and to provide an intuitive interface for others to explore the model and create their own unique variations. The *Components* were not created in the mirror image of their CATIA counterparts, but rather as their own interactive application. They were created with the notion that they would be modified to a point where the result could possibly yield an unrecognizable variation of the existing form of the Temple.

GC has proven to be a very powerful tool. However, it has failed to return results for models exceeding several thousand *Components*. This example reveals the need for tools that instill powerful ways of dealing with replication. Both CATIA and GC can only facilitate the editing of a single Temple Fin. Advances made to enhance usability perform quite well, scripts function appropriately, and editable variables perform their intended functions. Unfortunately the application cannot process the data quick enough to yield a timely response. This creates a somewhat static process, a slight variance of what already exists. Allowing the system to process modeling changes for an extended period of time eventually yields results. GC is simply incapable of performing this task dynamically.

What distinguishes the content of this chapter from other work is the extensive investment in usability. The nuts and bolts of scripts are virtually transparent to the user—as if the model was simply an interactive application in itself. There is a significant amount of work involved in this process. The advantages of having such a customizable tool—as with GC—is that the parametric controllers have the ability to be reshaped and customized to suit a user friendly aesthetic; unfortunately, this requires a considerable amount of intellectual investment. Buried within the *Structural Feature* is a script that searches the surrounding array for *instances* of the *Generative Component*. If it detects a duplicate member it skips the member and continues populating the remainder of the array. The concept of the array is a familiar idiom within Generative Components. It seems that GC's strength is in the use of such a familiar geometric construct.

The *Features* generated for this exposition test the *robustness* of Generative Components and challenge the idioms associated with a conventional process through the application of unique and complex *Features*. The content aims to create a parametric skeleton that can be used to explore variation in form.

D | FRAMEWORK

CHAPTER TITLE ILLUSTRATION *The pattern of bspline curves in this illustration were established by a script that was originally written by Dr. Robert Aish and used for demonstratiion purposes. The script draws a continuous 'bspline curve by points,' the script has been extended to array several copies of the resultant curve at random locations. The number of copies is controlled by a Global Variable.*

This chapter positions this thesis within the larger context of the state of development of parametric software applications and their relevance to potential users.

The Framework is an account of the current and historical development and use of parametric technologies in various disciplines. The Framework situates the body of this thesis within the context of current development initiatives and supplies a point of reference by which the Exposition better illustrates the lack of maturity in current parametric technologies as a design tool.

This chapter concludes and unifies the Application and Exposition chapters by relating the difficulties and concerns to a perpetually changing architectural process. These concerns include issues of complexity, workflow management and practical integration. The focus in this area of the industry appears to be in the documentation and translation of an idea and less about the advancement of the use of a tool for design space exploration. The advancement of these tools has mainly evolved out of their practical application and ability to reduce costs and streamline repetitive process, therefore the less profitable regions of process garner far less attention. According to survey participants this a familiar trend.

An investment in parametric research and development is spear-headed by a number of organizations,[1] and is growing exponentially. Software companies, institutions and private businesses are establishing a number of initiatives to help promote and advance the development of parametric tools.

Parametric tools have a significant impact on conceptual design processes. They allow the designer to decompose aspects of the design and precisely define building concepts and "practical fault lines." Encouraging the customization and exploitation of these tools will accelerate their development.

The survey attempts to addresses two impending issues: What is the significance of these tools? What are the current issues that affect development?

Survey Context

A parametric design survey was prepared and issued to prominent members of the *Smart Geometry Group*. This group is known for their vast contributions to the world wide community of parametric designers, developers and researchers. The purpose of the group is "to bring together the worlds of practice, education and research." [2]

The survey consisted of two parts: the first part was a set of general questions; the second part was two questions directly linked to the participants' own research interests. Additional or extraneous comments were accepted as non-required material. The principle of the survey was to gain valuable insight into the field of parametric research and design, including the current state of parametric tools, their historical context and the direction of future development.

Academia plays a large role in the development and testing of such systems, while the development of specific components stem from the needs of practice. Testing these parametric design tools in an academic setting fosters creativity and generates many unforeseen uses, "The aim of the advanced design technology theme is to foster industry and academia in using, adapting and creating new tools for design."[3] The use of parametric tools is becoming a necessity in practice. Such tools allow designers to focus more on design, and less on the risks commonly associated with the time, as with usage of *static* iterative design processes. Most participants admit that the field is still struggling through its infancy as it pertains to the realm of architecture, yet they agree that the process is rapidly progressing. Although precedents such as boat building, vehicle manufacturing, and aeronautical engineering, provide context for the development of these tools, they mislead by focusing on mass customization of components for production purposes instead of conveying the architect's desire for a discreet, customized solution.[4] Architects frequently create new uses for existing tools. They make use of software from the film and video game industries. However, the opportunity to design our own tools is a most desirable solution. It can be argued that this is not the responsibility of the designer, and rather the job of yet another consultant, however as designs become more complex, the process of documenting and constructing these designs will require more sophisticated tools. We should view this as an intellectual investment that enhances the quality of a body of work, and not as an intrusive obstruction of technology as it relates to the design process.

These tools allow us to explore and foster a closer relationship between concept and process. This allows architects to act as tool builders as well as designers: a task that has been characteristic of architects throughout history.

The following commentaries are accounts of the status and direction of the development of these tools as they relate to the advancement of the architectural profession and the process of architectural design.

Programme Participants

Dr. Robert Aish, *Director of Research, Bentley Systems, Incorporated*

Lars Hesselgren, *IT Director & Senior Associate Partner, KPF*

Axel Killian, *Dipl.-Ing., SMArchS, Ph.D Candidate, MIT*

Hugh Whitehead, *Partner, Foster & Partners*

Dr. Chris J.K. Williams, *Professor, Bath University*

Dr. Rob Woodbury, *Graduate Program Chair, Simon Fraser University*

GENERAL COMMENTARIES

The Queries

What disciplines have emerged out of the field of parametric research?

How has the field grown since its inception?

What aspects of parametric research require further development?

What factors are slowing down the progression of this research?

What are the conceptual and practical benefits of the tools developed from this research?

Robert Aish

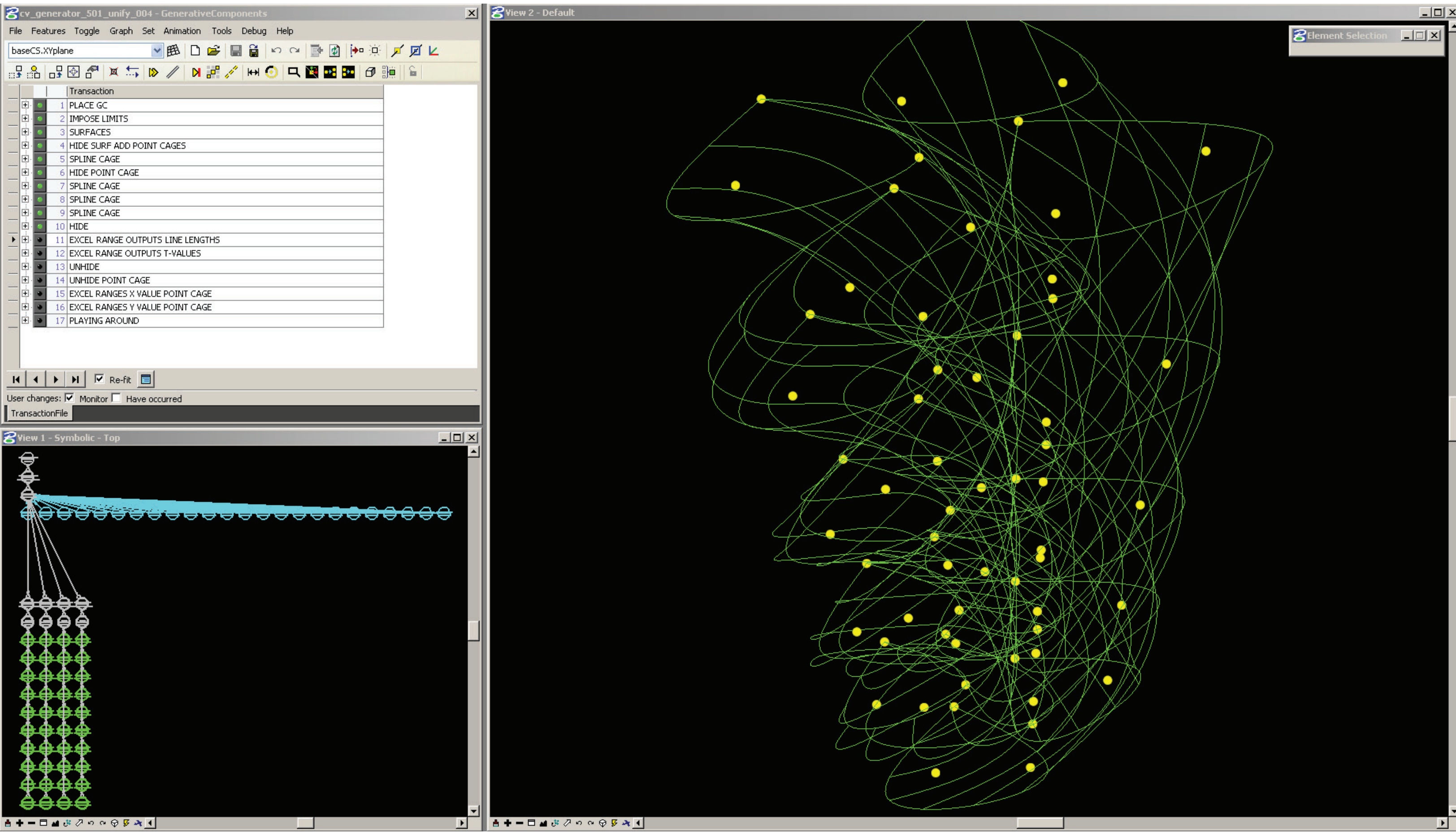
What disciplines have emerged out of the field of parametric research?
Applications such as GC which combine design tools and software development therefore encourage "design related software development," but that goes for many other disciplines, so really there is nothing new here. This combining of design and scripting was happening before GC with Rhino, Max and Maya, but with GC the whole platform is design oriented (rather than pure surface modeling—in the case of Rhino—or animation oriented—in the case of Max and Maya).

How has the field grown since its inception?
I have no metrics, but the number of schools and practices using GC (and other similar tools) is expanding rapidly.

What aspects of parametric research require further development?
How to teach it. I.E. how to train students and practitioners to combine design and algorithmic thought.

What factors are slowing down the progression of this research?
Teachers and design managers who have no such experience of algorithmic thought, or who are hesitant 'hands-on' users.

What are the conceptual and practical benefits of the tools developed from this research?
These tools essentially allow/force the designer to think. (Acid test: Does the student/teacher/practitioner/design manager think that this is a good or bad thing?)



4.01 Generative Components graphical user interface (GUI).

Lars Hesselgren

What disciplines have emerged out of the field of parametric research?
No new disciplines as such. Current disciplines such as mechanical engineering, architecture, structural engineering, environmental engineering have benefited in descending order.

How has the field grown since its inception?
The field grows by incorporating parametric techniques into existing CAD software it leaves some opportunity for new entrants but it is very small.

What aspects of parametric research require further development?
The whole system of understanding how a parametric model is structured needs far more development. Currently the content of a parametric model is a 'black box', its method of functioning only clear to the creator. And it doesn't help that the 'black box' is software specific.

What factors are slowing down the progression of this research?
Primarily the issue of competition between software vendors. On the upside however competition ensures that parametric tools are appearing in software used by all CAD users.

What are the conceptual and practical benefits of the tools developed from this research?
All buildings are systems. Systems that are openly declared can be verified in a more consistent manner. The declaration is in itself an intellectual tool for architectural thinking.

RIGHT *4.02 The Bishopsgate Tower, Photomontage view from the Tate Modern.*

Axel Kilian

What disciplines have emerged out of the field of parametric research?
I don't think there is a discipline emerging out of research around parametric studies. It is the other way round. Parametric descriptions of design problems have emerged out of the field of computation and design theory and engineering. Parametrics is not a novel concept but has a long history in different design domains. Its recent popularity is more a function of computing power and fabrication technology making buildings designed in this fashion more feasible.

How has the field grown since its inception?
See above—I don't think there is a field of parametric research per se but rather it is a part of many research areas in architecture and engineering.

What aspects of parametric research require further development?
The current implementation of parametric systems are far too rigid still to correspond to the design process. They tend to be implemented in a hierarchical fashion and allow very little flexibility in the definition.

What factors are slowing down the progression of this research?
The concept of Parametrics is too limited in its object oriented implementation, but of course powerful systems can still be built with that approach. But in order to reach the next level of complexity and design process support a more flexible and less hierarchical approach for capturing design intention is needed. This approach will certainly include parametric elements and concepts but probably be much more based on design exploration and variation on different levels than just the geometric one.

What are the conceptual and practical benefits of the tools developed from this research?
Design exploration and variation are the main benefits and the reuse of generalized constructs in different design context. The ability to build in a certain level of design intelligence in the componentry that constitutes the design assembly can help to integrate parallel domains like structural and performative design.

RIGHT *4.03 Carbon fibre tricycle.*

Hugh Whitehead

What factors are slowing down the progression of this research?
The issues of applying this borrowed technology in building tend to revolve around:

Scalability – When the desired level of detail is applied to complex buildings the resulting models tend to overpower current hardware capabilities.

Long Chain Dependencies – We must question whether full associativity is really required or even desirable? The effort involved both in setting up and maintaining associativity is not always justified or rewarded by gains in productivity or quality of performance.

Premeditation – If the idea is to 'encode design intent into models' this is easier to achieve later in the process or as a retrospective. It is not often a good starting point.

However the motive for any critique should be to recognise and transcend limitations. The potential of 'editable design' lies in empowering designers with new forms of language and notation.

What are the conceptual and practical benefits of the tools developed from this research?
In representational mode the designer has to freeze the early strategic decisions in order to progress to increasing levels of detail. This involves cyclic explorations but the early decisions can only be challenged if there is both time and resources to re-work the downstream details. In relationship mode the ability to populate an associative framework with adaptive components allows us to defer the decision-making process until we are ready to evaluate the results. We are now able to generate far more options than we are able to evaluate.

RIGHT *4.04 Swiss Re, London.*

Chris Williams

What disciplines have emerged out of the field of parametric research?
People naturally think parametrically, expecting a change in one thing to affect others. In particular comparing the sizes and proportions of things; you would expect a truck to have bigger wheels than a small car. Mathematics and, more recently, computing have always worked using parameters. Engineers and architects have also thought parametrically. What is new is the ability to use computers to automatically change lots of things as a few parameters are changed.

Obviously there is lots of work being done about the details of how all this is done, mainly by mathematicians and computer programmers. Sometimes it might be better to describe this work as 'development' rather than 'research'.

How has the field grown since its inception?
Massively. It's difficult to know quite how to reply, I suppose partly because it's difficult to pin down exactly what 'parametric design' really means. We are lucky in the building and civil engineering industries in that each project is a one-off, designed over a relatively short period by a relatively small design team.

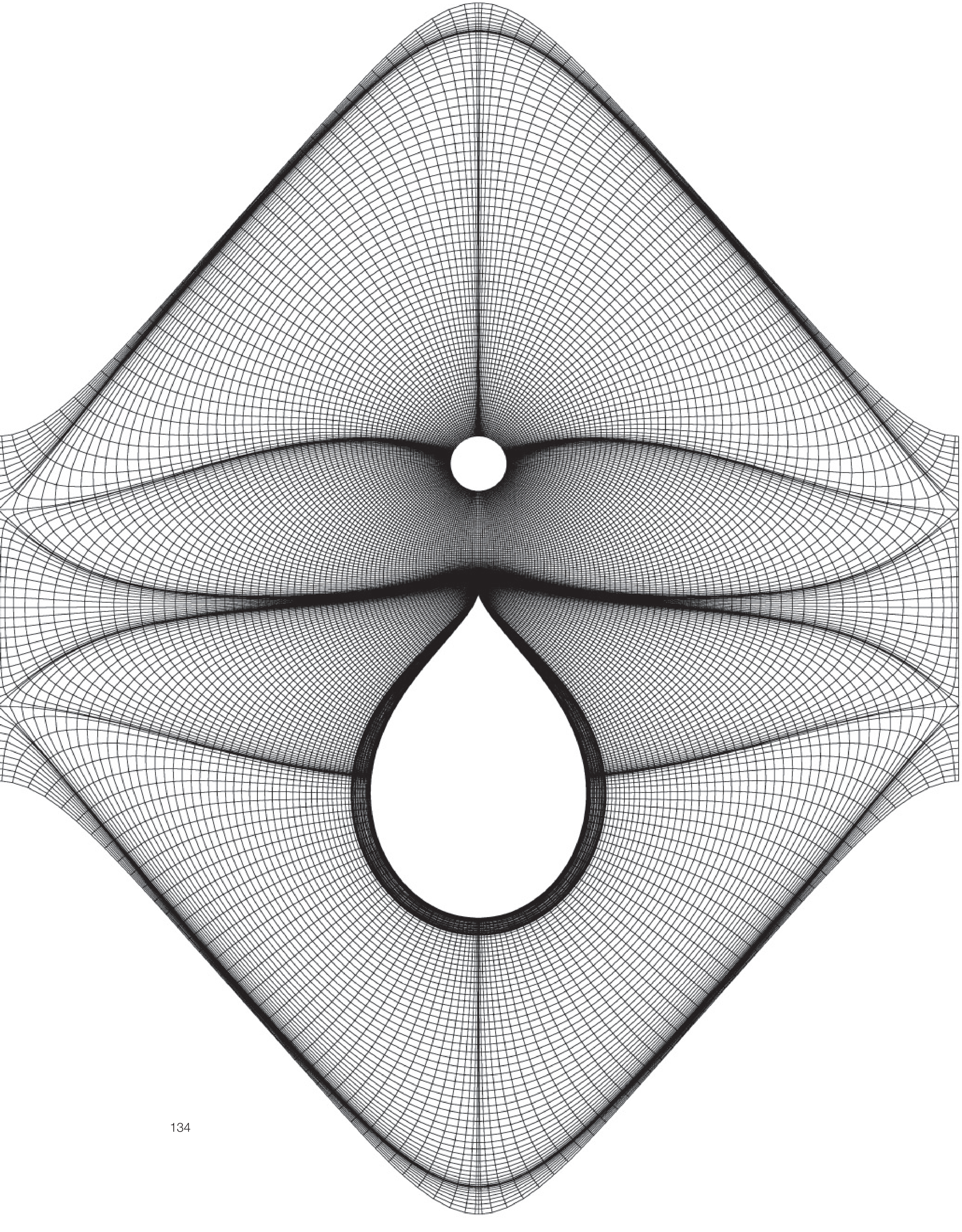
Compare this with, say, the Boeing 747 which was designed in less than 16 months and first flew in 1969. They still make it and it would be interesting to speak to the people who first designed it (who must be pretty old by now) and those who look after the design today changing thousands of bits.

So even though it is interesting to see what people like Gehry do with CATIA, one should really concentrate on Airbus, Boeing, Ford, etc.

www.boeing.com/history/boeing/747.html
www.boeing.com/commercial/747family/background.html
www.aventec.com/abmeth.html
www.boeing.com/commercial/777family/compute/compute4.html
www.practicalcatia.com/Ford.htm

What aspects of parametric research require further development?
Research tends to respond to a need. Much of the most interesting stuff is driven by film animation and this also includes technical things like fluid dynamics.

What are the conceptual and practical benefits of the tools developed from this research?
The benefits can be over emphasized. Are the objects designed now—buildings, bridges, airplanes, cars, ships - that much better than those designed without computers?



Rob Woodbury

What disciplines have emerged out of the field of parametric research?
It is actually the other way around. What disciplines have been applied in the building of parametric systems? The first CAD system was a parametric system (Sketchpad Sutherland 1963). Since then, computer science, mechanical engineering, mathematics, chemical engineering and operations research have been the main drivers. Parametric systems are relatively new in architecture, where they are fostering a reconsideration of many design issues and languages.

I interpret your question as being what new design applications have emerged out of parametric research. I would identify two vectors, one towards mass customization and the other towards manipulatable architecture. Mass customization identifies the complex of capabilities supported by the ever-decreasing gap between the price of mass-produced and customized items. Manipulatable architecture refers to buildings that move in some way. Parametric design is an enabling technology for both.

How has the field grown since its inception?
Its inception was in computer science, most application has been in mechanical and aerospace engineering. In those fields, parametric systems are a mainstay. It is at the beginning of what looks like to be a growth curve in architecture, but it is at the beginning.

What aspects of parametric research require further development?
My view is that the interfaces are primitive, the useful methods of work using such systems largely unknown and that discrete parameterization is where the big gains and the greatest difficulties lie.

It takes a long time to do anything in a parametric system. I believe this is partly due to the design of interfaces for such systems. We simply do not have good tools for composing objects and for seeing reasonable ranges of parameterization.

The higher-order ways in which people use parametric systems are poorly understood. This will take some serious social science research to uncover. Parametric systems most easily support change when change is smooth. When jumps are made, for instance when

LEFT *4.05 Study of shell structure.*

a new assembly is introduced depending on some parameter, current systems are weak in both representation and interface. Discrete parameterization sharply reveals the need for a design space representation, that is, an explicit representation of the space of alternative designs considered by the user (or users).

What factors are slowing down the progression of this research?
Scale of industry. In architecture it remains small.
Graduate programs with appropriate faculty expertise and courses of study.
A clear body of work around the issue.
Of course, money. But that problem is always there.
In other words, the research field is young and people in it can make rapid progress.

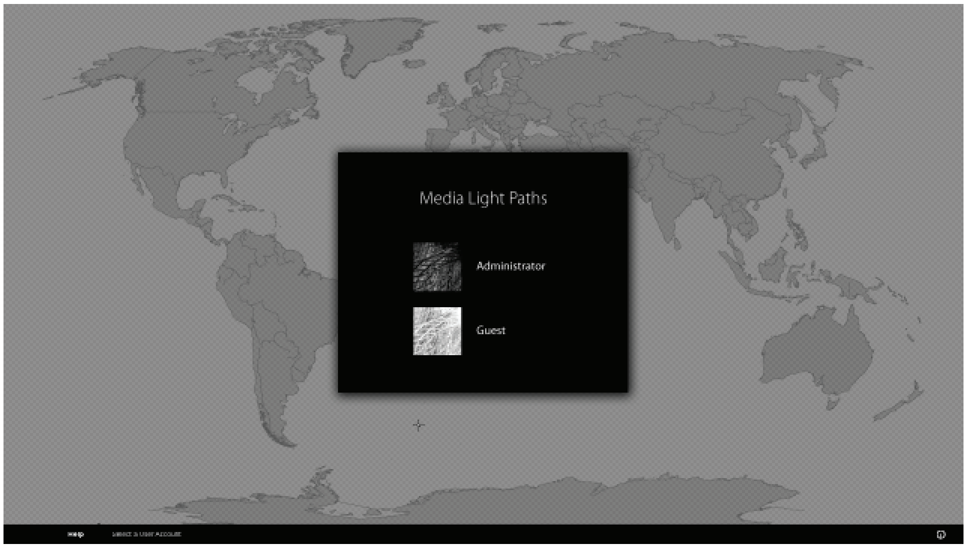
What are the conceptual and practical benefits of the tools developed from this research?
I include an excerpt from a recent proposal:

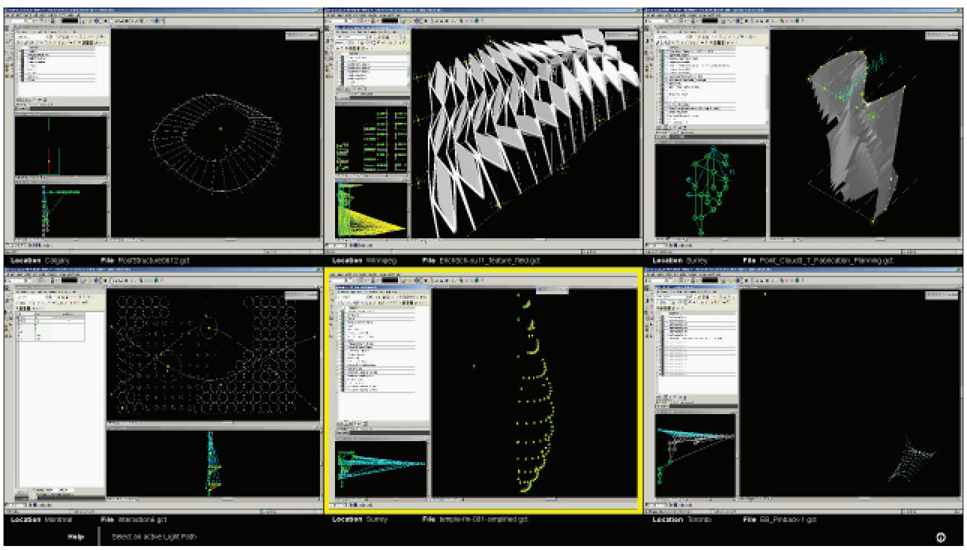
> Design work is transiting to digital media and computer-based tools (Mitchell & McCullough 1994, Eastman 1999). This has profound effects on both design work itself and the products of that work (Aish and Woodbury 2005). For example, the design of the new roof on the courtyard of the British Museum depended utterly on digital representation and simulation (Williams 2001). In such work digital data are not limited to designers, for example, in buildings they are transferred between architects, structural engineers, and fabricators. Firms employing new digital tools can gain real advantage in national and international markets. A major obstacle to progress is a lack of highly qualified personnel who understand both design work and the new tools. Another obstacle is an incomplete understanding, in both academia and industry, of how new design media can transform both design process and outcomes. The aim of the advanced design technology theme is to foster industry and academia in using, adapting and creating new tools for design.

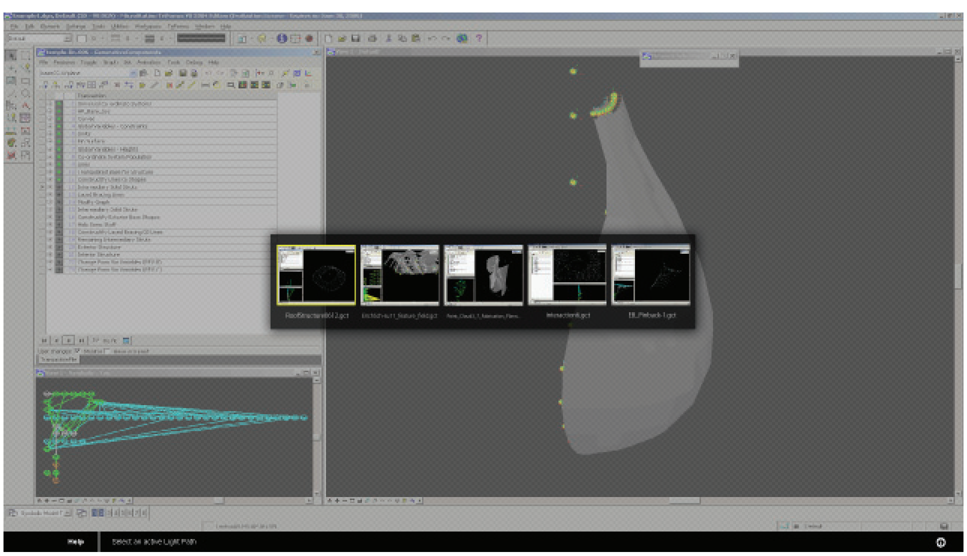
Conceptual benefits include a disciplined understanding of a form of change. Since design is the process of making proposals for change, such is important in the field.

Practical benefits include the opening of new formal and construction possibilities. These create economic advantages for early adopters.

RIGHT *4.06 Captured frames from a multimedia presentation of the Media Light Paths demonstration.*


Media Light Paths
Administrator
Guest





RESEARCH COMMENTARIES

Robert Aish

What is the conceptual programming model for Generative Components?
It is pretty rare to find a building which is realized as a single discrete object. Normally we are considering assemblies of components which, at intermediate levels of aggregation, form identifiable sub-systems. While these components may be pre-defined, or the subsystems may follows established industry conventions, there are increasing opportunities for each design to use mass customization and digital fabrication to define project specific components. The question then is: how do we break down the total building concept into sub-systems and components? What are the conceptual or practical 'fault lines' which might suggest this decomposition? There may in fact be multiple decompositions, some to be used in the conceptual, form finding phase, and others for realization and fabrications which, for example, might impose dimensions constraints associated with different materials or fabrications processes. What is certain, is that developing and refining compositional strategies is a key aspect of design skills. There is a tremendous advantage in using computational design tools which directly support the idea of 'composition' and which allow these strategies to be developed and tested.

What scope of precision does Generative Components give to the designer?
Design has been described as making inspired decisions with incomplete information. True, we may use prior knowledge, we may even think we understand the causalities involved, but what really matters is exploration: of new forms, of new materials, and speculation about the response to the resulting effects. Essentially, this exploration has its own dynamics, involving intuition and spontaneity, and without which there is no design.

But of course we all know that this is not the whole story. Design is different to 'craft'; to directly 'making' or 'doing'. It necessarily has to be predictive in order to anticipate what the consequence of the 'making' or 'doing' will be. Therefore we inevitably have to counter balance our intuition with a well developed sense of premeditation. We have to be able to reason about future events, about the consequence of something that has not yet being made. There is always going to be an advantage if this reasoning can be achieved with a degree of precision.

PREVIOUS *4.07 Temple model built during the early conceptual stages of design.*

Lars Hesselgren

How does your research contribute to practice?
Until recently the research has been completely project focused. The result is seen in numerous buildings, such as the Bishops gate Tower, which simply would have been almost impossible to design with conventional means.

In-house expertise is continually building and at some point becomes indistinguishable from in-house software development.

Is there a polemic between practice and research in the industry? Should the two be one in the same (please advocate your response)?
There is a polemic but it is disjoint. Much (most, almost all in fact) academic research does not impinge on practice. The most common link is by researchers moving into practice using their research ideas and methodologies.

There are two distinct areas: architectural design and software design. These two have developed into different skill sets and software design has a larger impact because it spans multiple disciplines and may redefine the professional boundaries.

Architectural design operates on the level of ideas. In the academic world they often seem very remote from commercial practice, but there is currently nowhere else to 'dream professionally' (no risk of actually building the dream!)

I would say there is a place for 'professional dreaming' within practices; but it is hard because benefits are vague and long term. In many cases they turn into a branch of marketing.

How do significant studies in the field of parametric design relate to the ongoing battle between research and its relationship with architectural practice?
The academic world has its own concerns and aims which are not appreciated by practice.

The race to publication which serves as almost the sole guideline of career advancement is particularly debilitating. It leads among other things to 'cultural ghettoes' where the insiders talk in code to each other.

Some of these ghettoes are significant and will lead to changes in behaviour. Most are simply internal talking shops designed to enhance the members' status.

Practices judge by a different yardstick, in its own way as obnoxious. Built buildings and marketing publications are their yard sticks.

Ultimately it is a battle of ideas, some of which become embedded in technology and then become mundane. Other ideas are pervasive and have long-term effects, think architectural styles.

Axel Kilian

How do you feel the effects of application programming in design will affect the culture and ideologies commonly associated with traditional process?
I think it is about adopting the process to the design problem at hand and how the digital realm can be integrated back into existing and developing design processes in what has been called the post-digital era in design. It could lead to a variant of craft that is not based on personal expertise and tradition but rather on the ruse of knowledge and fast adaptation in complex design process with the use of digitally represented and integrated design.

What are your thoughts about the infiltration of parametric technology into the realm of architecture? What aspects are deemed a valuable asset? Which are not?
Parametric concepts and thinking is already present in design, the development of software environments to support these existing concepts allows them to be pushed much further and subsequently new aesthetic and conceptual approaches to emerge. The variation and exploration of clearly defined highly constrained geometry-centric design problems profit a lot from it.

Quickly changing, conceptually driven abstract design problems do not benefit from the current batch of parametric software that is more centered at the dimensional descriptive intention of paramterics.

Hugh Whitehead

How does the intense pressure of working in a project driven environment affect your working methodology?
At Foster and Partners the Specialist Modelling Group provides in-house consultancy to project teams at all stages from concept design to detailed fabrication. Although we provide Tools, Techniques and Workflow these are developed in the reverse order. Starting with the formulation of the problem the first step is to propose an appropriate workflow. Within this frame of reference suitable techniques are tried and tested in different combinations. The results then form the brief for the development of custom tools that are tested by the design team in a continuing dialogue. Custom tool building ensures that rationale becomes an integral part of the design concept.

Tools are developed for use by the designers who are directly involved in the specification and testing cycles which ensures a relationship that is more synergetic than symbiotic. By working in parallel with many design teams we are ideally placed to encourage cross-fertilization of ideas and techniques. Tool building becomes a cumulative process. As well as capturing design intent we also distribute expertise.

This is achieved by taking a modular approach to building tools. Operations that can be written in a generic form are taken out to a function library, so that tools become progressively easier to build and maintain or adapt to new requirements.

How have parametric tools changed they way in which you troubleshoot design issues? Have they altered your process specifically in any way?
The SwissRe building forced us to address the problem of how to design and produce programmatic details. At each floor the rules are always the same but the results are always different. At the same time even if every plan, section and elevation could have been drawn, this still would not adequately describe the design intent even for tender purposes let alone construction. The building stands as a classic example of an associative framework providing

a context for adaptive parametric components, so that fabrication follows a consistent dialogue between structural and cladding node geometry.

The designer is in charge of the rehearsal but the contractor is responsible for the performance. We are limited in what we can build by what we are able to communicate. Many of the problems we now face are problems of language rather than technology. The experience of SwissRe established successful procedures for communicating design through a Geometry Method Statement.

Complex geometries involve very large parameter sets that are impossible to control by direct manipulation. With buildings like the Beijing Airport, which has a double curved roof that is three kilometres long, the approach was to develop control mechanisms that can be driven by law curves. Law curves control 'rate of change' and can be geometric (as graphs) or algebraic (as functions). By representing higher derivatives as curves or even surfaces, complex behaviour can be achieved with simple manipulation. For example the law curve for a spiral is a straight line, representing the linear relationship between offset and rotation angle. Manipulating the line dynamically generates a set of offsets that can be used to drive other parametric sub-assemblies.

Efficiency is about achieving 'more with less' in terms of the resources used for implementation. However at the concept stage we aim to do 'less thinking, but with more intelligence'! In order to reduce the solution space we also aim to produce less options but with more creativity. Design is not just a Darwinian process of natural selection that can rely entirely on brute-force computing. However the way we represent design ideas is becoming changed by the way we implement them. The power of the sketch is being augmented by the power of the schematic – which is the minimal digital representation of an idea or concept.

Chris Williams

How do you establish what tools to use when optimizing form?
There is always some sort of algorithm—a set of rules so that the computer can generate lots of data. I prefer to write my own rules, but in general designers have to use rules written by someone else—a bit depressing.

Do mathematical precedents play a large role in your work or do you simply refer to them as the language by which the form is established?
Rules have to be expressed in terms of mathematics and Boolean relations—if this and that [are] true then do whatever. Computer programs give the impression of some sort of intelligence, but it is simply the application of thousands of lines of code.

Rob Woodbury

What key factors do you hold in high regard when designing an application? What conceptual form do these factors take?
My answer is a caricature of what makes real estate valuable (location, location and location). Clarity, clarity and clarity.

I design applications for academic reasons—to make points that are not being made in industry. Therefore I am less concerned with broad functionality than with the clear presentation of new ideas. I look for strong ideas that provide new insight into areas of design. Each of the applications I have built that others have used has provided something different that was not available in other tools. I implement to explain, so clarity is a foremost goal.

How can building simulation and design space emulation be enhanced through parametric technology?
In a nutshell, parametric technology is one of the possible engines for a design space explorer. In fact, the ideas of creating variation (parametric technology) and managing variation (design space exploration) are highly complementary.

DISTILLATION

This survey provides a historical context of the subject defined by individuals that have worked with parametric technologies over many decades. The historical context they provide is one that returns to the days before this technology ever existed; when planes, boats and cars were built by "traditional methods." Chris Williams poses the question, "Are the objects designed now...that much better then the ones designed without computers?"[5] This thesis argues that it is not a matter of whether they are superior or inferior, but rather the impact the technology has on the process that allows us to achieve a final result. Certainly there are some practical advances that have come from being able to use parametric technology within the context of the design process but as Dr. Williams states it would be wrong to suggest whether or not this makes the new better than the old. It simply allows designers to experiment in ways that were never thought possible in the past.

These tools have significant impacts on conceptual design processes, they allow the designer to break down and precisely define building concepts and "practical fault lines." The designer can decompose aspects of the design throughout the many phases of the design process. Defining the concepts and decompositions can be handled in many ways. These exercises generate several conceptual alternatives.

The introduction of these tools in architecture raises a number of developmental issues and constraints. The current tools are centred on descriptive methods of design rather than areas that are less defined (areas that do not require dimensional driven data). The development of architectural parametric tools needs to integrate more expressive modeling methods, and perhaps the ability to defer parametric constraints between certain objects. The largest developmental issue is the lack of integration in education and practice, the tools need to be taught and explored by a larger number of design researchers and professionals.

This thesis adds to the current development of parametric technology by making particular contributions to tools within the realm of parametric research.

An Author's Response to the General Queries

What disciplines have emerged out of the field of parametric research?
Disciplines have not emerged out of the field of parametric research, however, many jobs have. Many of the participants in this survey (myself included) agree that this question should be re-written to ask: How has the advent of parametric research evolved and progressed through the needs associated with practice and scholarly research initiatives?

How has the field grown since its inception?
There is no field specifically designated to advancing parametric technologies, there are several areas of study which incorporate the notion or ideology of what it means for an object or component to be parametric. The definition of parametric technology has broadened and incorporated into products that could not be entirely defined as being parametric. The maturation of this technology has occurred due to the needs and requirements of those that require easier ways of working through iterative process, the field benefits and grows through the use and development of new and existing products.

What aspects of parametric research require further development?
Passing on the knowledge! Convincing those in practice and academia that this is an area worth pursuing. Teaching the fundamental concepts of what the software does and can do is a major hurdle at the moment. There are a lot of options that exist, but there are many opinions on which products are worth teaching. The mature products can cost hundreds of thousands of dollars, the relatively new products require further development to be seen as a practical alternative. It is a catch twenty-two; businesses don't want to play around with development software, and the software companies don't want to fund a project that doesn't have a large install base. An open language that is platform independent that could allow users to easily transfer their work between applications would be both incredibly useful and ground breaking.

What factors are slowing down the progression of this research?
Mainly competition amongst various software vendors and funding. A competitive software economy is useful in the sense that it stimulates rapid growth and maturation of the products which compete, however it also creates delays. Certain products get pushed back because they are lacking essential features that exist in within competing products. Funding is limited in this realm of research, although now that this technology is at the hands of so many we might see this change. The most pressing factor is education. How do we pass the torch to the next generation? Right now, this seems to be one of the most pressing concerns—getting more people involved.

What are the conceptual and practical benefits of the tools developed from this research?
The conceptual benefits can be viewed in one of two ways, by their inclusion in conceptual design and their effects on process. Conceptually these tools allow the designer to distill their intentions throughout the design concept, literally allowing the concept to drive the model. The process is altered by the inherent nature of change due to altering how a model is changed. The practical benefits are clear; many iterations can be generated, due to investing more time at the outset, which has the potential to increase the amount of detail generated and therefore reduce the amount of change and intervention required at the back-end of the project. Another obvious practical benefit is that we can experiment with these tools and invent new ways of deriving and controlling form. The things we create with these tools are in no way better, they are merely different or previously unexplored.

An Assessment of the Queries

The participants' responses to the General Queries fit appropriately within the limits of this thesis. They all address how a historical process is being enhanced by new technologies. If the notion that "people naturally think parametrically"[6] is true, one would have to ask why is parametric technology such a novel concept? It can be argued that this is due to the recent availability and influx of powerful modeling tools.

While the concept of having a single dynamic model is an appealing objective for most, it has the ability to drastically affect performance. "Is full associativity even required or desired?"[7] Typically it is not, and this is due to a number of factors; certain features or aspects of a design stay constant (i.e. windows, doors, etc.), in which case it does not make sense to create a large amount of overhead by making these components dynamic. Factors such as building envelope, square footage and overall form might all change on a consistent basis. The key is finding a healthy dynamic that satisfies both static and dynamic building components. "The potential of 'editable design' lies in empowering designers with new forms of language and notation."[8]

While many parametric tools exist none can be considered exemplary or perfect by any means. Almost all participants agree that the concept or language of what duties a parametric tool performs must be altered drastically. The current state of the tools is "far too rigid" to be able to fully satisfy all requirements of a dynamic design process. Parametric systems inherently demand definition, without defining relationships you cannot create. This is an issue directly related to language, notation and hard-wired limitations in the tools themselves. Of course there is always a means of by-passing hard-wired limitations through scripting and other forms of development, but unless one has access to the nuts and bolts of a tool, these features can never be altered. The participants agree that in order to create tools that are teachable a careful study of comprehensible user interface design is crucial.

The traditional model of the designer is being slightly altered by this technology. At Foster and Partners it is almost entirely altered in the sense that individuals who were once engineers or designers have become computer scientists. Although this may seem alarming, it is—at its core—beneficial to all designers. Allowing the individuals who use the tool to design it makes perfect sense. "By working in parallel with many design teams we are ideally placed to encourage cross-fertilization of ideas and techniques. Tool building becomes a cumulative process. As well as capturing design intent we also distribute expertise."[9] The synergetic relationship that already exists in process is not altered but rather enhanced by parametric exploration.

The figures shown in the Application component of this thesis were generated in iterative phases. Daily revisions of the geometry were captured and exported which maintained accurate and up to the minute versions of design alterations. Images were in turn generated and returned, prompting additional changes. This process could not have existed without the "cross-fertilization" of many individuals across many disciplines—designers, fabricators and engineers. Parametric processes ensure that a wealth of data is generated promptly and accurately.

Many architecture firms are becoming aware of the benefits of this technology in practice and are also aware that it is still in its infancy. Hugh Whitehead acknowledges this when he asks, "is full associativity even required?" Lars Hesselgren, Axel Kilian and Dr. Woodbury maintain that the entire ideological construct of parametric software technologies needs to be re-evaluated and re-constructed to suit architects and designers. Chris Williams argues that the benefits of this technology "can be over emphasized;" currently, this may be true. All participants agree that parametric technology is going to drastically enhance the way designers draw and communicate their ideas to others.

Placing the Framework in Context

This thesis positions itself within the realm of the development of parametric technology through an exposition. The potential of such technology is explored through the original contributions made in the Exposition and are meant to enhance and add to the Framework which places the work in a larger context.

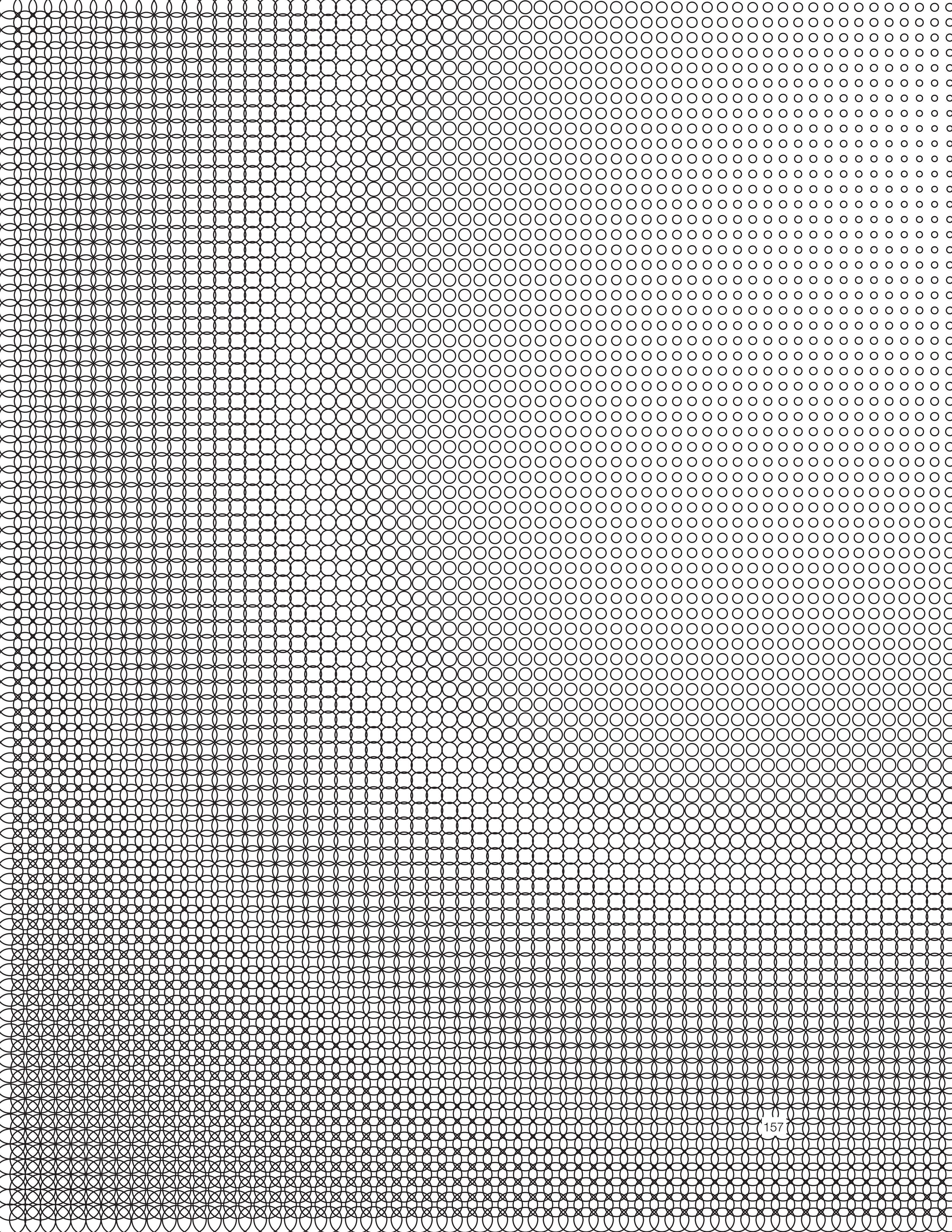
The participants' responses provide a clear view of the status of this technology as it relates to architecture—it is young and requires further development. Although the software has yet to mature, it is already having an impact on projects that employ its use.[10] Both KPF and Foster and Partners use this technology to convey their drawings to third parties. They use it as an explanatory tool rather then a tool solely used for drawing.

The survey participants agree that the most prevalent area of development should be in establishing new conceptual relationships between the features within the tools. This involves investing time in new software features as well as in an easier way to utilize these features. Developing three dimensional parametric models consists of careful planning exercises. In order for this technology to become more popular the fundamental inner workings of parametric software needs to be changed to suit a dynamic design process.[11] The current state of the tools satisfies the process of documentation and translation, and rightly so, as these areas of the architectural process facilitate the physical construction of an architectural idea. For designers to truly embrace the tools they need to feel that their creative concerns have been addressed as well. User friendly interfaces, the ability to create digital sketches, conceptual planning modules are but a few features that are non-existent within parametric tools.[12]

The documentation strengths of these tools have become the focus of vendors developing parametric software technologies. The most effective way of influencing the direction of the technology is to team up with a vendor to receive exclusive access to pre-release technologies. Experimentation with these technologies throughout a "live" process is both frightening and enlightening. Without this kind of experimentation the technologies have little chance of developing into holistic, feature packed and dynamic alternatives.[13] Integrating an explorative developmental process into the architectural process creates many synergetic design driven relationships.[14]

Experimentation fosters innovation. The Bahá'í Temple, the content in this thesis, and the work of the participants in this survey are a testament to the relevance of this statement as it relates to an architectural process. The research of the participants in this chapter is advancing the state of the tools. The participants are members of the Smart Geometry Group, a network of practitioners and researchers whose objective is to unite the worlds of practice and research, to foster innovation and to promote the use of experimentation in process.

This content provides a working framework and context in which to place the thesis. It contains both current and historical precedence that associate the relevance of the Application and Exposition sections within an architectural parametric process.



E | CONCLUSION

EVALUATION

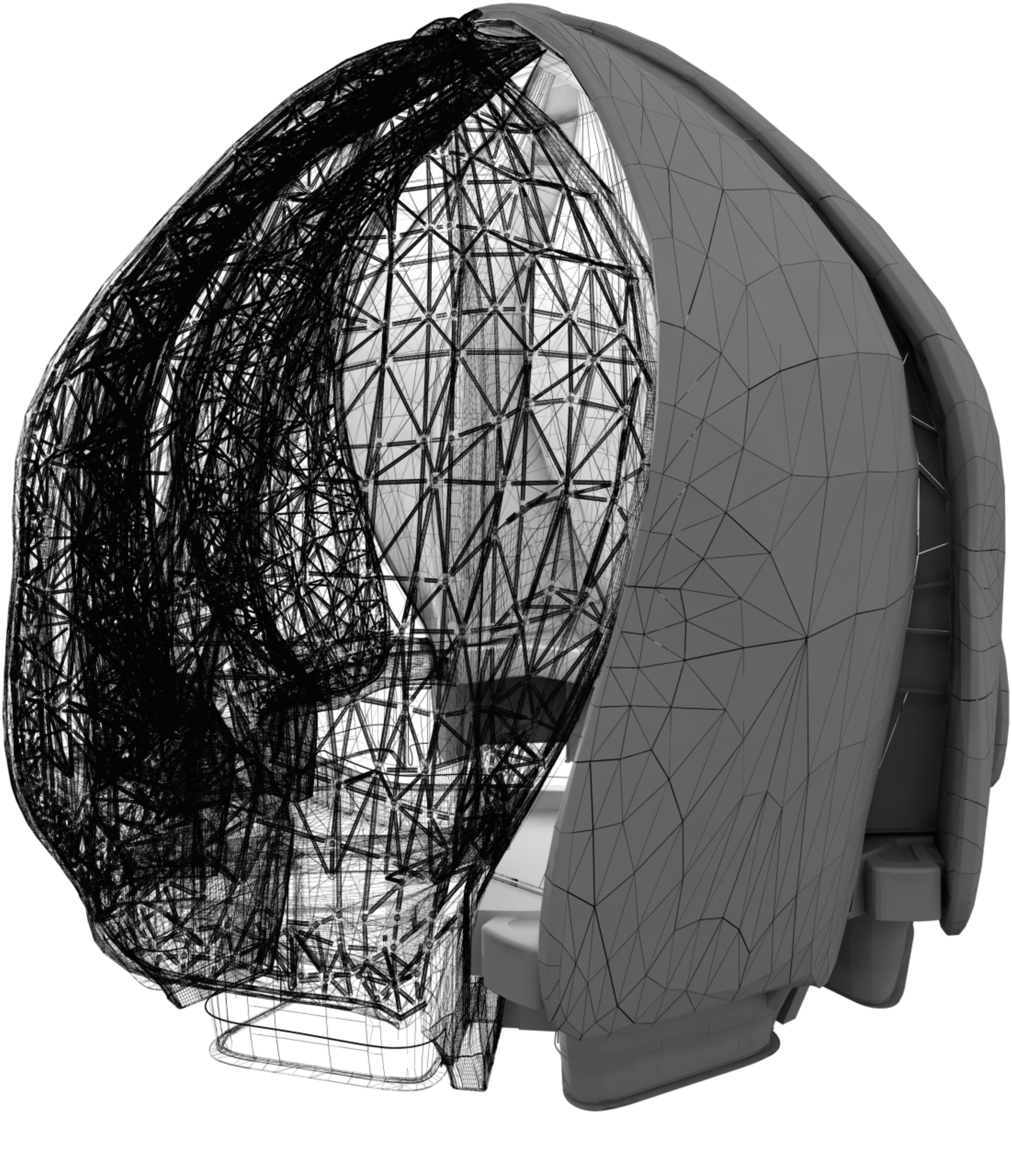
CHAPTER TITLE ILLUSTRATION *The radii of the circles in this illustration are determined by a set of invisible controllers: the circles sit on top of an array of points; several invisible points are scattered within the array; and the lengths of the lines that connect the scattered and arrayed points drive circle radii. The line lengths are then plugged into a function that returns an average length. This produces a unique pattern within the array. Using a single length creates linear incrementations in circle radii.*

This thesis evaluated a parametric design tool and focused on the implementation of this tool in an architectural process. It addresses the burgeoning state of the tools that are currently available, and exposes an ever growing number of essential requisites that require further development in order for parametric modeling to become a viable alternative to static modeling.

The three components of this thesis—Application, Exposition and Framework—organize the research by introducing the context of the issue, exploring the issue through the use of a discrete tool, assess the state of the implementation of parametric tools in research and industry, and most importantly, describe the benefits of parametric tools.

The chapter entitled Application, is a case study of the Bahá'í Temple for South America. This chapter precedes the Exposition to objectify the content that is used throughout the thesis. It touches briefly on the implementation of CATIA and how the implementation relates to the process Hariri Pontarini Architects (HPA) is using to complete the project. This component of the thesis defines the relationships and components of the building that are used to establish the parametric model created in the Exposition.

The exposition of Generative Components (GC) is introduced to create a correlation between the process of design and the implementation of parametric tools. It focuses on describing the process of implementation for each component. Several unique Features and Scripts are used to express Generative Components' cumbersome user interface, open scripting capabilities and powerful array techniques. The contents of the GC model are used in conjunction with the GC development process to expand and troubleshoot the *Features* that will be bundled with the software upon its release. The exposition asserts the validity of parametric drawing and modeling processes, and introduces their ability to enhance creative process.



The thesis concludes with a survey entitled Framework, the survey places the thesis within the context of industry and research initiatives. It offers brief, pertinent historical references and documents how each participant integrates parametric technology into his own research. The Framework has been assembled to corroborate the process of its selected participants and to present examples that exemplify their efforts. The contents of this chapter are analyzed and distilled in relation to the content of the thesis to dispel any obscurities.

This thesis explores and evaluates the implementation of parametric design technologies within the process of design and through the exploration of Generative Components. It touches on the current state of parametric design tools, it investigates technology still undergoing development (Generative Components), and it places these findings within the context of comments made by individuals implementing these technologies in their own research and practices.

The thesis delivers an analysis of the Bahá'í Temple by Hariri Pontarini Architects, an exposition of Bentley's Generative Components, and a survey of the parametric research initiatives that are taking place in academia and practice.

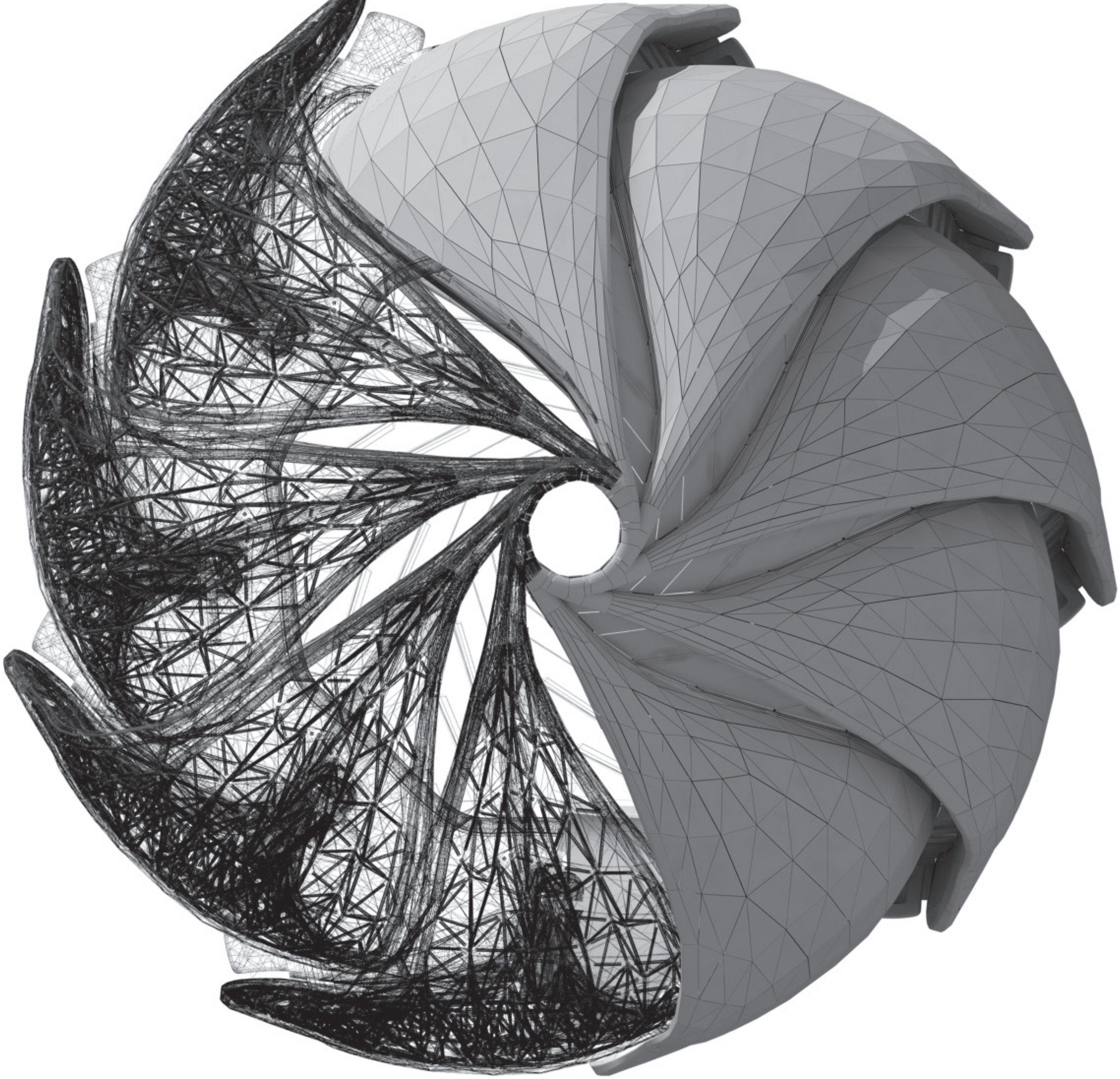
LEFT *5.01 Rendering that represents the dichotomy between structural complexity and formal simplicity.*

EXPLORATION

The following text presents additional areas of exploration and attempts to speculate on ideal features of a parametric design tool. The text invites interest in future areas of parametric development.

I have focused on Generative Components (GC) as an alternative to CATIA. This process involved maintaining the formal qualities of the original model while reconstructing the data using GC. CATIA can be used to model any object. The process of modeling a pre-existing object is explained through the process of documentation. GC was developed in the eye of the designer. Its strength lies in the ability to create model based representations of conceptual ideas. This aspect of parametric technology is highly under-developed. The current tools require the cumulative construction of modeling components, every component of a parametric model has a dependency, if the dependency is removed then all other objects based on that dependency are removed as well.[1] The software is doing what it is meant to do because the developers are writing it do so. They are writing software that stays true to the notion of what it means for an object to truly be parametric, they are creating software that allows the user to create models of complex interdependencies. This is expressed in the Exposition chapter of this thesis. Each Component description is written in the manner in which it was modeled, this exploits the hierarchical nature of the process. Objects are built upon objects, which relate to, and drive other objects represented within the same network. GC is an effective tool for creating parametric models of considerable scale and complexity. Further research into the areas of optimization and scalability might yield a process that would allow the construction of more complicated models. Although the limitations are not within GC itself but rather the platform in which it was built on.

The content covered in this thesis documents current conventions and notes future avenues of research. Areas such as the cognitive aspects associated with parametric tools and the act of design, software development, sharing parametric projects (allowing multiple users to interact with the same model), and the use of parametric software as



LEFT *5.02 Rendering that represents the dichotomy between structural complexity and formal simplicity.*

a means to design responsive building envelopes and structures—architecture that physically manipulates itself based on its response to various environmental factors. Mechanical engineers have been using this technology for years, mainly in robotics. Architects and designers are just starting to tap into the possibilities. Phil Ayres, an English architect associated with the Bartlett School of Architecture, is using Autodesk Inventor for his case study of the Kiedler Forest in Northumberland, UK.[2] He has devised digital parametric representations that are manipulated by data that has been gathered by environmental changes in the forest. His demonstrations show the digital model responding to the variable environmental conditions. This reveals that it would be possible to fabricate a full scale version of the digital model, instilling the software mechanics that would allow the constructed version to adapt itself as well. There are many areas of parametric research and development that have yet to be realized, in order to effectively explore these areas we need to establish the tools that facilitate these explorations.

The rigid mechanics of parametric software need to be redefined in order for it to be considered a viable alternative to traditional CADD (Computer Aided Drafting and Design) applications. A critical concern for future development is the exploration of an intuitive and flexible modeling interface, currently this does not exist, the powerful features of existing tools is buried beneath miles of programming code. A weakness of these tools is their requirement to initiate "dependencies."[3] The ability to create "lazy" dependencies—dependencies capable of becoming independent—would be useful.

NOTES

Context

1. Axel Kilian's fourth response in the General Queries outlines the minimal capacity of building applications upon an object oriented platform: Kilian, Axel. *Digital Process Commentaries*. (Toronto: Mark Cichy, 2006), 1.

2. An online repository of papers relating to fabrication, digital design techniques and general CAD technologies can be found at: Martens, Bob, and Turk, Ziga. "Cumincad." SciX Open Publishing Services. http://cumincad.scix.net/cgi-bin/works/Home (accessed 08/29, 2006).

3. The following workbook has some very useful CATIA tutorials for beginners: Cozzens, Richard. *CATIA V5 Workbook: Releases 10 & 11*. 5th ed. Cedar City, Utah: Schroff Development Corporation, 2003., I-1.

4. This document is mainly a source for tutorials, but it also introduces the concepts behind GC and why it was created: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 2-5.

Application

1. The following reference contains a repository of teachings and writings on the Bahá'í Faith: Bahá'í International Community. "The Bahá'Ís." Bahá'í International Community. http://www.bahai.org/ (accessed 08/30, 2006).

2. Extensive information for members of the Bahai Fatih can be found by visiting a respository managed by the Universal House of Justice at: Universal House of Justice. "Nine-Pointed Star, the History and Symbolism." UHJ Unpublished. http://bahai-library.com/?file=uhj_nine_pointed_star#s3 (accessed 08/30, 2006).

3. This website contains information specific to the Temple for South America: Hauser, Robert, and Fierling, Mark. "The Bahá'í Temple for South America." Bahá'í

International Community, http://temple.cl.bahai.org/html/en/slide1.htm, accessed 09/04, 2004.

4. This website contains information specific to the Temple for South America: Hauser, Robert, and Fierling, Mark. "The Bahá'í Temple for South America." Bahá'í International Community, http://temple.cl.bahai.org/html/en/slide1.htm, accessed 09/04, 2004.

5. This book is a comprehensive resource of CAD/CAM technologies and how these technologies can be used to manufacture building components: Schodek, Daniel L. *Digital Design and Manufacturing : CAD/CAM Applications in Architecture and Design*. Hoboken: John Wiley & Sons, 2005, http://www.loc.gov/catdir/toc/ecip0419/2004014940.html; http://www.loc.gov/catdir/description/wiley042/2004014940.html., 10, 23, 29.

6. The introduction of this workbook clearly defines aspects of the CATIA interface: Cozzens, Richard. *CATIA V5 Workbook: Releases 10 & 11*. 5th ed. Cedar City, Utah: Schroff Development Corporation, 2003., I-9.

7. This website has a very flexible steel calculator, it allows you to choose from a list of common steel sections or create your own section; it will also allow you to choose different metals to compare weight and costs: "Steel Tools and Conversion Tables." MEsteel.com. http://www.mesteel.com/start.htm (accessed 02/28, 2006).

Exposition

1. This source is solely distributed in a digital Word document and is included with every installation of GC. It is a manual that explains the user interface and features of the product. To locate the file simply browse to where you installed the software : Kilian, Axel, and Robert Aish. "Generative Components."Bentley Systems, Inc., Exton.

2. The Smart Geometry Archive is an excellent online resource for GC related content. You can post and download scripts. This enables a self taught approach to the material: Chen, Victor, and Woodbury, Rob. "SmartGeometry Archive, Architectural Design with Computational Design Tools." Canadian Design Research Network. http://smartgeometry.designscience.ca/ (accessed 04/20, 2006).

3. Kroll's research occured at a time when computers were first being introduced to architects. This book is an excellent historical reference that helps place the current state of tools in context: Kroll, Lucien. *The Architecture of Complexity*. London: Batsford, 1986., 91.

4. The introduction of Chapter 8 glosses over user defined components and how they are created in GC: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 8-87.

5. A comprehensive manual on GC: Kilian, Axel, and Robert Aish. "Generative Components. "Bentley Systems, Inc., Exton., 2.

6. This document is useful for those just starting with GC. First published at ACADIA in 2004, it contains some very useful tuorials: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 2-5.

7. Dr. Aish's introduction in this manual describes GC as an exploratory tool for designers, he goes on to address the pracicality of this realm of exploration: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 2-5.

Framework

The Digital Process Commentaries are a set of requested unpublished research surveys that were submitted to gain information on the state of the research and application of parametric tools in industry. The full commentaries are presented in their entirety in the Framework chapter of this thesis.

1. Organizations contributing to parametric research and development include Bentley Systems Inc., Gehry Technologies, ACADIA, The Canadian Design and Research Network, CumInCAD, The Smart Geometry Group, Foster and Partners, KPF, and many other notable organizations and corporate institutions.

2. Whitehead, Hugh. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 1.

3. Woodbury, Rob. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 2 .

4. Williams, Chris. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 1.

5. Williams, Chris. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 2.

6. Williams, Chris. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 1.

7. Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006), 1.

8. Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006), 3.

9. Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006), 3.

10. Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006), 4.

11. Woodbury, Rob. *Digital Process Commentaries* (Toronto: Mark Cichy, 2006), 2 .

12. Kilian, Axel. *Digital Process Commentaries*. (Toronto: Mark Cichy, 2006), 1.

13. Aish, Robert. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006, 2.

14. Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006), 2.

Development

1. The introduction to Chapter 5 of this manual describes the ideology of dependencies within GC: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 5-55.

2. This book covers topics on electronic art and performance installations, research in cell structures and natural systems, and the design of interactive buildings: *Responsive Architectures : Subtle Technologies 2006*. Cambridge, Ont.: Riverside Architectural Press, 2006., 48,49.

3. Dependancies are inherent by-products of parametric tools. The dependant connection between two objects has to exist in order for the objects to exist. Disassociating dependancies breaks the parametric chain and doing this initiates a rebuilding process. Currently it is possible to re-assign dependancies, but doing so deletes the geometry that is left disassociated. Dr. Aish defines dependancies in the following manual: Aish, Robert. "Generative Components - Introduction. "Bentley Systems, Inc., Exton., 5-55.

INDEX OF ILLUSTRATIONS

IMAGE CREDITS & PROCESS

Photographs

1.04	**10/11**	Mark Cichy.
4.03	**129**	Chair: Enrique L. Garcia/Axel Kilian/Patrik Künzler; Design: Axel Kilian; Frame: Axel Kilian, Peter Schmidtt; Wheel: Peter Schmidtt.
4.04	**131**	Nigel Young, Foster and Partners.
4.06	**139**	Mark Cichy.

Illustrations

2.23	**43**	Mark Cichy. *CATIA surfaces and solids, translated into mesh geometry, rendered with visualization software, modified with 2D compositor, sections imported from third party CAD software and composited in illustration software.*
2.24	**44**	Ibid.
2.25	**45**	Ibid.
2.26	**46**	Ibid.
2.27	**47**	Ibid.
2.28	**48**	Ibid.
2.29	**49**	Ibid.
2.30	**50**	Ibid.
2.31	**51**	Ibid.
2.32	**54**	Mark Cichy. *Screenshot modified in 2D compositor.*
2.33	**54**	Ibid.
2.34	**55**	Ibid.
2.35	**57**	Mark Cichy. *CATIA surfaces and solids, translated into mesh geometry, rendered with visualization software and modified with 2D compositor.*
2.36	**57**	Mark Cichy. *Screenshot modified in 2D compositor.*
2.37	**57**	Ibid.
2.38	**58**	Mark Cichy. *CATIA surfaces and solids, translated into mesh geometry, rendered with visualization software and modified with 2D compositor.*

3.50	**113**	Mark Cichy. *Sketches scanned and modified with 2D compositor.*
3.51	**113**	Ibid.
4.01	**125**	Mark Cichy. *Screenshot modified in 2D compositor.*
4.02	**127**	KPF and Cityscape. *Mesh geometry, rendered with visualization software and modified with 2D compositor.*
4.05	**134**	Chris Williams. *DXF generated from executing program code edited in illustration software.*
5.01	**162**	Mark Cichy. *CATIA surfaces and solids, translated into mesh geometry, rendered with visualization software and modified with 2D compositor*.
5.02	**166**	Ibid.

BIBLIOGRAPHY

Cognitive Understanding

Augé, Marc. *Non-Places : Introduction to an Anthropology of Supermodernity*. London ; New York: Verso, 1995.

Debord, Guy. *The Society of the Spectacle*. New York: Zone Books, 1994.

Hughes, Howard C. *Sensory Exotica : A World Beyond Human Experience*. Cambridge, Mass. ; London: MIT Press, 1999.

Contemporary Design

Bonet i Armengol, Jordi. *L'Últim Gaudí : El Modulat Geométric Del Temple De La Sagrada Família = the Essential Gaudí : The Geometric Modulation of the Church of the Sagrada Família*. 1a ed. Barcelona: Editorial Pòrtic, 2000.

McQuaid, Matilda. *Shigeru Ban*. London: Phaidon, 2003.

Otto, Frei, Bodo Rasch, Gerd Pfafferodt, Adelheíd Gräfín Schönborn, and Sabine Schanz. *Frei Otto, Bodo Rasch : Finding Form : Towards an Architecture of the Minimal*. 1995th ed. Germany: Werkbund Bayern; Museum Villa Stuck; Axel Menges, 1995.

Tschichold, Jan. *The New Typography : A Handbook for Modern Designers*. Weimar and Now. Vol. 8. Berkeley: University of California Press, 1995.

Digital Design & Implementation

Responsive Architectures : Subtle Technologies 2006. Cambridge, Ont.: Riverside Architectural Press, 2006.

Aish, Robert. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.

———. "From Intuition to Precision." in Cumincad, 2005.

Beesley, Philip, ed., Nancy Cheng ed., and Shane Williamson ed. *Fabrication: Examining the Digital Practice of Architecture*, edited by Philip Beesley, Nancy Cheng and Shane Williamson. 1st ed. Toronto, ON: Coach House Press, 2004.

Beesley, Philip, ed., Vincent Hui ed., and Michael Stacey ed. "Digital Fabricators." Cambridge, ON, Coach House Press, November 2004, 2004.

Escuela Técnica Superior de Arquitectura de Barcelona. *Genetic Architectures = Arquitecturas Genéticas*. Santa Fe, New Mexico: Lumen Books/SITES Books, 2003.

Friedman, Mildred S., Michael Sorkin, and Frank O. Gehry and Associates. *Gehry Talks : Architecture + Process*. New York: Rizzoli, 1999.

Hauser, Robert, and Fierling, Mark. "The Bahá'í Temple for South America." Bahá'í International Community. http://temple.cl.bahai.org/html/en/slide1.htm (accessed 09/04, 2004).

Hesselgren, Lars. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.

Kilian, Axel. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.

Kroll, Lucien, and Peter Blundell-Jones. *The Architecture of Complexity*. London: Batsford, 1986.

Leach, Neil, Chris Williams, and David Turnbull. *Digital Tectonics*. Chichester, U.K. ; Hoboken, NJ: Wiley-Academy, 2004.

Lynn, Greg. *Fold, Bodies & Blobs: Collected Essays*, edited by Stephanie C. Jaeger. Amsterdam: Idea Books, October 2000.

———. *Animate Form*. New York: Princeton Architectural Press, 1999.

Maeda, John. *Design by Numbers*. Cambridge, Mass.: MIT Press, 1999.

Martens, Bob, and Turk, Ziga. "Cumincad." SciX Open Publishing Services. http://

cumincad.scix.net/cgi-bin/works/Home (accessed 08/29, 2006).
Rahim, Ali. *Contemporary Techniques in Architecture*. Profile. Vol. 155. London ; New York: Wiley-Academy, 2002.
Stacey, Michael, Beesley, Philip, and Hui, Vincent, eds. *Digital Fabricators*. Toronto, ON: Coach House Press, 2004.
Watanabe, Makoto Sei. *Induction Design : A Method for Evolutionary Design*. The IT Revolution in Architecture. Basel ; Boston: Birkhäuser, 2002.
Whitehead, Hugh. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.
Williams, Chris JK. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.
———. "Dr Chris JK Williams." University of Bath. http://staff.bath.ac.uk/abscjkw/ (accessed 04/06, 2006).
Woodbury, Rob. *Digital Process Commentaries*. Toronto: Mark Cichy, 2006.

Pattern Recognition

Batty, Michael. *Cities and Complexity : Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*. Cambridge, Mass.: MIT Press, 2005.
Chaplain, M. A. J., J. C. McLachlan, and Gurdev Singh. *On Growth and Form : Spatiotemporal Pattern Formation in Biology*. Wiley Series in Mathematical and Computational Biology. New York: Wiley, 1999.
Feininger, Andreas, and University of Richmond Museums. *Structures of Nature : Photographs*. Richmond, Va.: University of Richmond Museums, 2002.
Thompson, D'Arcy Wentworth. *On Growth and Form*. An abridg ed. Cambridge, England : Cambridge University Press, c1961: .
Wolfram, Stephen. *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002.

Technical Documents & Manuals

"Steel Tools and Conversion Tables." MEsteel.com. http://www.mesteel.com/start.htm (accessed 02/28, 2006).

Addington, D. Michelle, and Daniel L. Schodek. *Smart Materials and New Technologies : For the Architecture and Design Professions*. Amsterdam ; Boston: Architectural Press, 2005.

Aish, Robert. "Generative Components - Introduction."Bentley Systems, Inc., Exton.

Bechtold, Martin, ed., Kimo Griggs ed., Schodek, Daniel L., ed., and Marco Steinberg ed. "New Technologies in Architecture:Digital Design and Manufacturing Techniques." Graduate School of Design, Harvard University, President and Fellows of Harvard College, October 2000, 2000.

Bechtold, Martin, ed., Kino Griggs ed., Schodek, Daniel L., ed., and Marco Steinberg ed. "New Technologies in Architecture:Digital Design and Manufacturing Techniques II & III." Graduate School of Design, Harvard University, President and Fellows of Harvard College, January 2003, 2003.

Bhattacharyya, S. P., H. Chapellat, and L. H. Keel. *Robust Control : The Parametric Approach*. Prentice-Hall Information and System Sciences Series. Englewood Cliffs, N.J.: Prentice Hall PTR, 1995.

Chen, Victor, and Woodbury, Rob. "SmartGeometry Archive, Architectural Design with Computational Design Tools." Canadian Design Research Network. http://smartgeometry.designscience.ca/ (accessed 04/20, 2006).

Cozzens, Richard. *CATIA V5 Workbook: Releases 10 & 11*. 5th ed. Cedar City, Utah: Schroff Development Corporation, 2003.

Greenwold, Simon, and Allen, Edward. "Active Statics." Massachusetts Institute of Technology. http://acg.media.mit.edu/people/simong/statics/data/

(accessed 10/05, 2006).

Kilian, Axel, and Robert Aish. "Generative Components."Bentley Systems, Inc., Exton.

Liberty, Jesse. *Programming C#*. 3rd ed. Sebastopol, CA: O'Reilly, 2003.

Magnenat-Thalmann, Nadia, and Daniel Thalmann. *Modelling and Motion Capture Techniques for Virtual Environments : International Workshop, CAPTECH'98, Geneva, Switzerland, November 26-27, 1998 : Proceedings*. Lecture Notes in Computer Science. Vol. 1537. Berlin ; New York: Springer-Verlag, 1998.

Schodek, Daniel L. *Digital Design and Manufacturing : CAD/CAM Applications in Architecture and Design*. Hoboken: John Wiley & Sons, 2005, http://www.loc.gov/catdir/toc/ecip0419/2004014940.html; http://www.loc.gov/catdir/description/wiley042/2004014940.html.

Temkin, Aron, ed. "Evolving Tools: Digital Fabrication in Architectural Education." Cambridge, ON, Coach House Press, November 13, 2004, 2004.

Webb, Barbara, and Thomas R. Consi. *Biorobotics : Methods and Applications*. Menlo Park, Calif.: AAAI Press/MIT Press, 2001.

GLOSSARY

active or passive modeling environment	The dynamic effects of change can be activated or deactivated, passive updates occur only when initiated by the user.
algorithm	Mathematical equation, either single or string.
application models	Conceptual software models.
associativity	They way in which Parts, Products and files are connected.
bidirectional	The act of submitting and receiving information.
BSpline Curves	A curve composed of control points and a knot vector—with a defined degree or order of curvature.
BSpline Surfaces	A surface composed of control points or BSpline curves—with a defined degree or order of curvature.
CATIA	Constraint based parametric modeling application.
complex curves	Surfaces curved on more then one axis.
component	A single object or group composed of a part or parts that make up a particular building feature.
componentization	Breaking up a single object into smaller parts.
Components	Objects compiled from Features within Generative Components—can also be a Feature itself.
Component System	A Generative Component with single or nested Components.
computationally driven	Systems that are altered by variation in mathematical equations and GC Script code.

constraining	Employing variable limits amongst modeling components.
controller	Any Component (geometric or otherwise) that can be used to change the values of variable or GC.
customized software features	High-level features, not low-level programming code.
data-sets	Dense bodies of information. These can be statistical, geometric or numerical.
design space exploration	Using digital design tools as a method of exploring spatial relationships.
development cycle	Yearly production timelines that bring products to the consumer or end-user.
direct offset	Offsetting an object in a straight line of projection.
embedded data-driven constructs	The act of instilling information that has the ability of driving or piloting design decisions.
end struts	The ends of the strut that connect to the ball joint.
essential design tools	Basic Features required for visually modeling three-dimensional components—points, lines, arcs, etc.
expressions	Simple mathematical formulas that return a usable value.
fabricator	The individual or organization converting the three dimensional model into data that can be used to generate physical scale models.
facetization	Partitioning a surface into coplanar or non coplanar shape facets—may be quadrilateral or triangular.

Feature	Also referred to as a Generative Component—it is a functional software component that is either built into GC or is created by the user.
form	Three dimensional geometry.
framework	The base feature set of the Generative Components application.
Generative Components	Script/Transaction based parametric modeling application.
geometric	Geometric data created with a three dimensional modeling application.
geometric versus application model	A model made up of three dimensional data as opposed to the conceptual outline of a software application.
Graph Function	A Generative Components Feature that facilitates the use of customized C# programming scripts.
Graph Variable	A GC Feature that may be used to control any value within a Generative Component through the Graph Variables interface.
greater accuracy	Increased variation throughout the conceptual design process, increases the relevance and precision of the model as it relates to the designers intentions.
grid lines	Paths used to optimize the location of structural components—do not have to be linear.
information management	Organizing drawing and related data in a coherent way and legible manner.

intelligible	Clear, concise, and easy to understand.
intelligent	Aware of the other Components in the immediate vicinity.
instances	Duplicate items, identical in every way.
internal code	Platform specific code—GC Script.
internal structure	Enclosed components supporting building.
Law Curve	A type of curve that can be used to generate variables that have the ability to drive values within a GC.
library	A collection of items that have been made available to the user to help facilitate the use of the tool.
machining	Fabricating digital data into physical objects—traditionally machining involves working with metals.
macros	Small scripts that automate repetitive tasks.
manufacturing technologies	Technology that allows the designer to submit digital data so that it can be machined into physical objects.
milling	Fabricating digital data into physical objects using a multiple axis router.
modeling methodologies	The act of establishing ideological software constructs; inventing conceptual frameworks.
network	Groupings or arrays of Generative Components.
normal	A vector which is perpendicular to a surface.

object-orietned	A type of programming structure, the ability to categorize objects within containers—a combination of code.
Parts	A type of component modeled in CATIA. A portion or segment of a Product.
parametric design tools	Software tools that facilitate the use of variables and equations to modify objects on global and discreet scales.
physical geometry	Geometry that has been drawn by executing a Transaction in Generative Components.
platform	The object oriented .NET programming language developed by Microsoft for Windows.
pre-scripted elements	Components that have been created by GC Scripts.
pre-release technologies	Also known as Alpha or Beta testing, involves testing products at their early stages, when they are highly unstable and feature lacking.
Product	The root assembly object in CATIA. A collection of Parts.
Product Lifecycle Management (PLM)	Increasing the longevity of a product through digitally detailing every aspect of its construction, making it easier to modify and release updated products.
programming	Application independent code—may be dependant upon operating system.
radians	A circular unit of measure.
robust	A flexible and powerful geometric entity.
robustness	An application that is robust exhibits indestructible

	qualities of persistence, an application that is un-crashable.
scalable content economy	Finding ways of conserving computing resources throughout efficient three dimensional modeling processes.
scalar process	Breaking down the process into several more manageable parts, can be used to solve issues of complexity.
scripting	Also referred to as the Graphical User Interface.
Script Transactions	Hold the data that is used to draw the three-dimensional geometry to the display.
Smart Geometry Group	A group of professionals and academics dedicated to advancing parametric technology through the unification of practice and academia.
Specification Tree	The Specification Tree contains the history of tools and processes used to create a Part or Product.
static	Refers to CAD data that is not parametric.
streamlined components	Building components that have been resolved and refined—absolving building details of any unforeseen eccentricities.
structure	The elements of the building that maintain its stability.
Sub-Components	Nested Generative Components—Components within Components.
surface offset variable	The Graph Variable created to control the distance of internal and external surfaces from the point of registration.

Symbolic Graphs	A symbolic representation of the parametric relationships created in a Generative Components model.
system	Interconnected objects.
top-level Components	The root Component of a Feature that contains nested Generative Components.
Transaction File	The native file format for files created with GC.
tri-axial dynamism	Adding perceptual interest to the model by exploiting all three dimensions.
user interface enhancement	Using the tools avaiable in Generative Components to create visual controllers that add to the UI experience.
variations	Different revisions of three dimensional geometry that are related to the same conceptual idea.
visual interface	Also referred to as the Graphical User Interface.
Workbenches	Toolkits for three dimensional modeling—the way CATIA organizes user feature sets.
workflow	The process of establishing a sequence of events that is optimized for maximum efficiency.
Z-vector	The direction vector of the z-axis.