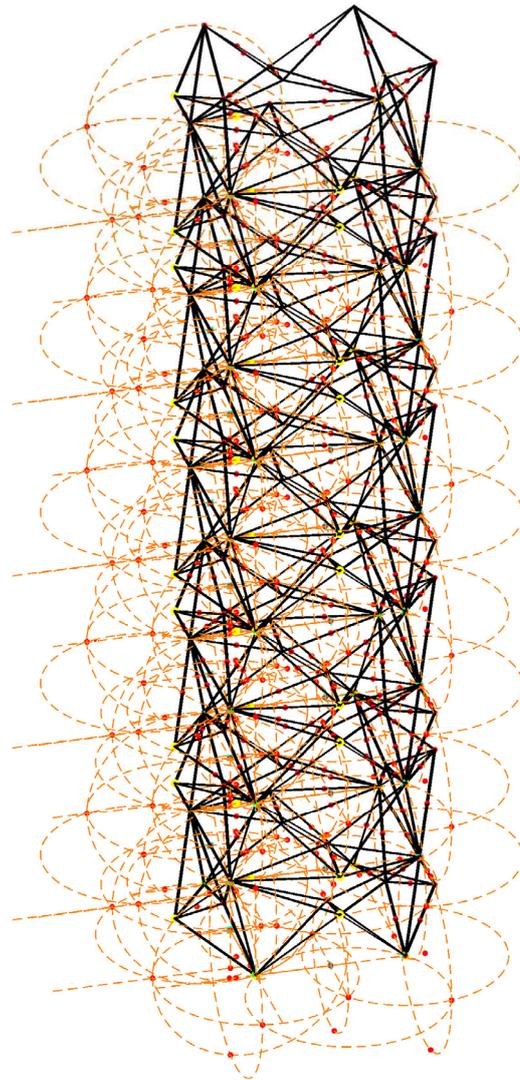


Solving constraints within a graph based-dependency model by digitising a new process of incrementally casting concrete structures

Bengt Cousins-Jenvey



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

Bartlett School of Graduate Studies

University College London

September 2008

I, Bengt Cousins-Jenvey , confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

The mechanisation of incrementally casting concrete structures can reduce the economic and environmental cost of the formwork which produces them. Low-tech versions of these forms have been designed to produce structures with cross-sectional continuity, but the design and implementation of complex adaptable formworks remains untenable for smaller projects. Addressing these feasibility issues by digitally modelling these systems is problematic because constraint solvers are the obvious method of modelling the adaptable formwork, but cannot acknowledge the hierarchical relationships created by assembling multiple instances of the system. This thesis hypothesises that these opposing relationships may not be completely disparate and that simple dependency relationships can be used to solve constraints if the real procedure of constructing the system is replicated digitally. The behaviour of the digital model was correlated with the behaviour of physical prototypes of the system which were refined based on digital explorations of its possibilities. The generated output is assessed physically on the basis of its efficiency and ease of assembly and digitally on the basis that permutations can be simply described and potentially built in reality. One of the columns generated by the thesis will be cast by the redesigned system in Lyon at the first F2F (file to factory) continuum workshop.

9,760 Words

Contents

Abstract	3
Contents.....	4
Illustrations.....	6
Acknowledgements	8
Introduction	9
Background	12
Aims and Objectives	20
Methodology	24
Physical Prototype One	24
Form	24
Frame.....	25
Faces.....	26
Standard Components.....	28
Primary Points	28
Possible Meeting Points	29
Possible Secondary Points and Possible Planes.....	30
Real Points and Real Planes	31
Scripted Components	34
User Generated Features	38
Combinations of User Generated Features.....	40
Results	43
Discussion	51
Digital Characteristics of the System	51
Physical Characteristics of the System.....	53
Other Approaches.....	58
MicroStation.....	58
Dimension Driven Design (DDD).....	59
Limitations	61

Further Work.....	62
Conclusion.....	63
Bibliography.....	65
Appendix I.....	67
Appendix II.....	70

Illustrations

Figure 1 - Timber forms for a chimney with cross-sectional variations (Wynn & Manning 1926).	13
Figure 2 - Forms to describe a domed roof (Wynn and Manning 1926).....	14
Figure 3 - Column Sections generated by the interference between two helicoids (Burry, 1993).....	16
Figure 4 - The making of a column model based on a system invented by the model makers which allows the sequential counter-rotation of the zinc profile over the setting plaster of Paris (Burry, 1993).	17
Figure 5 - Multiple states of Prototype One.....	25
Figure 6 - Two stacked instances of the physical prototype.	26
Figure 7 - One instance with visible construction geometry.....	28
Figure 8 - Primary points (blue), secondary points (red) and meeting points (green).....	28
Figure 9 - Possible locations of meeting points created by the intersection of two spheres.	30
Figure 10 - Four possible secondary points created by five intersecting circles.....	31
Figure 11 - The creation of a real secondary point by two intersecting circles.	32
Figure 12 - The creation of the other secondary points.	32
Figure 13 - Completed construction of one instance (subject to rotation).	33
Figure 14 - The “levels of granularity” within the overall design of the system and the tensions between them.	38
Figure 15 - Two instances of the system combined ‘by meeting points’	40
Figure 16 - Two instances of the system combined ‘by secondary points’ . ..	41
Figure 17 - Four instances of the system combined ‘by meeting points’ and ‘by secondary points’ to show the three different types of interstitial parts. .	42
Figure 18 - Table of eliminated and generated forms as simple two parameter descriptions.....	43
Figure 19 - X11Y11 drawings and renders.....	45

Figure 20 - X11Y15 drawings and renders.....	45
Figure 21 - X11Y19 drawings and renders.....	46
Figure 22 - X15Y15 drawings and renders.....	46
Figure 23 - X15Y19 drawings and renders.....	47
Figure 24 – X19Y19 drawings and renders.....	47
Figure 25 - Ten instances of the system generated from X15Y19F0.....	48
Figure 20 – X15Y19F2 elevations.....	49
Figure 21 – X15Y19F4 elevations.....	49
Figure 22 - X15Y19F8 elevations.....	50
Figure 23 - X15Y19F0, X15Y19F0 and X15Y19F0 renders.....	50
Figure 24 - The refined physical prototype.....	53
Figure 25 - Workshop photos of machine assisted fabrication.....	54
Figure 26 - The first instance of the second prototype with locating washers.....	55
Figure 27 - Tightening the tension screws to fix the first instance.....	55
Figure 28 - Attaching faces to the first instance.....	56
Figure 29 - Matching the primary points of the secondary instance with the secondary points of the first instance.....	56
Figure 30 - Locating the second instance of the system.....	57
Figure 31 - Tightening the tension screws to fix the second instance.....	57

Acknowledgements

I would like to thank Alasdair Turner for introducing me to coding and my supervisor Sean Hanna for enthusing about my ideas, clarifying my thoughts and suggesting practical solutions to many of the problems I encountered.

I would also like to thank Eddie Jump for his early engineering perspective and the expertise of Abi Abdolwahabi, Richard Grimes and Stephen Pugh who all made physically modelling the system possible.

My friends, particularly Laura Whateley and Ross Masood, provided support, insightful comments and lateral thinking.

Introduction

The main focus of this study is the production of a digital model and algorithm from a prototyped, physical process of construction. This is proposed as an alternative to starting with a digital model and generating complex structures which, without sufficient grounding in reality, cannot be post-rationalised and easily built. Approximating a digitally conceived structure or resolving its construction retrospectively is neither as elegant nor as efficient.

Instead of designing a complex structure and then considering how it might be built, or whether it is buildable at all, the challenge is to have the foresight to design and build a prototype of an adaptable module with broad architectural applications. This is potentially restrictive; digitally modelling the permutations of the tessellations of this prototype will not only test and document what is possible, but also provide crucial information about the construction of its different aggregations. It is also important to show that a cyclical process of refinement, with digital tests continually informing new physical prototypes, might tend towards solutions tailored to specific design problems.

This might apply to the assembly of modular structures or, more challengingly, to the mechanisation of a process of construction. Such a specific purpose is crucial to establish the criteria by which the system will be assessed beyond aligning the behaviour of a digital model with the physical prototype of it. Using the digital model to inform the practical application of a refined physical prototype is a better test and was formulated from a critical appraisal of jump and slip forms. These methods of casting concrete which reduce the explicit monetary and environmental impact of the

temporary formwork by permitting extrusions and other repetitive structures to be cast incrementally, are however decades old. Although some variation in the cross sections of cast structures has been achieved by these techniques (Richardson 1977), studying the proposed process would seem to be validated by an ambition to incrementally cast even more complex forms and to examine the organisation of the formwork that might permit a digitally controlled metamorphosis of this process.

This new system should cast complex structures which belie the set of repetitive parts and, like many elements of the Expiatory Church of the Sagrada Familia, the simple geometric rules that produce them (Burry, 1993). A balance or equilibrium must be achieved: too few stages and the formwork (as a consequence of its perceived adaptability and minimalism) might deflect or fail completely, but too many casts and the process may be too chaotic or too slow to be tenable. Research into the self-assembly of aggregating structures (Sass, 2004) and integration of emerging fabrication techniques might counteract the problems perceived with the latter scenario, by expediting and simplifying construction of the system.

The challenge is to not only develop the mechanisation of this systematic construction process, but to build and explore its possibilities digitally. All elements of a dynamic are likely be equal, with a change to any one of them demanding that all of the others are updated in response. Hierarchy and chronological procedure are not important, just the constraints which govern the static positions the mould can adopt. These different, fixed permutations are important, not how it moves between them. Modelling the process of constructing different instances of these machines however, demands a clear understanding that each instance of the machine is stacked upon, and therefore becomes dependent upon, the last. The chronology of construction

and resulting dependencies are crucial if the digital model is to inform the real system.

The problem is that the simplest solution to these contradictory properties of the system is to employ two disparate types of model that have not yet been integrated. Some elements of the structure are equal and have bi-directional relationships (those from which each instance is constructed), but some are clearly dependent and have directional relationships (those from which subsequent instances are constructed). It has been hypothesised that constraint solvers will enable bi-directional relationships to be established within a graph-based dependency model. This study hypothesises that a system with bi-directional and directional aspects can be modelled digitally without employing constraint solvers, if the model replicates the specific process by which it is constructed in reality.

Making the world's available resources serve one hundred percent of an exploding population can only be accomplished by a boldly accelerated design revolution to increase the present performance per unit of invested resources.

Buckminster Fuller 1970

...less is more...

Robert Browning 1855

Background

A good ambition is to increase the output of a process by reducing the waste produced and improve its efficiency. A better ambition is to defy what is logically possible and reduce the input, but increase the output: thereby doing more with less. These are the properties of emergence. Numerous examples of natural and artificial systems display emergent properties and there are indications that it is possible to imagine structures - inspired by both biology and computation - which also display these characteristics. Rather than magically producing more from less, new methods can seem emergent *in comparison* to existing processes. The illusion might be achieved by reassessing the complexity of the output and discarding properties which are surplus to requirements, introducing new machines and technologies or combining and reusing components.

Very efficient processes of casting concrete have been employed for decades. Slipforming minimises the amount of formwork required to cast repetitive

structures by continuously (slipform) or intermittently (jumpform) casting the required profile (Fig. 1).

The actual slipforming technique consists of filling a set of forms which are continuously or intermittently raised or moved to generate the required structural profile. The length of forms, rate of travel and stiffening time of the concrete are so arranged that the concrete which emerges from the bottom opening of the form achieves sufficient strength to support the load of the form and the fresh concrete above as the slide continues (Richardson, 1977, p.247).

In order to critique these processes, it is important to compare their utilisation and design with standard methods and materials to understand their efficiency.

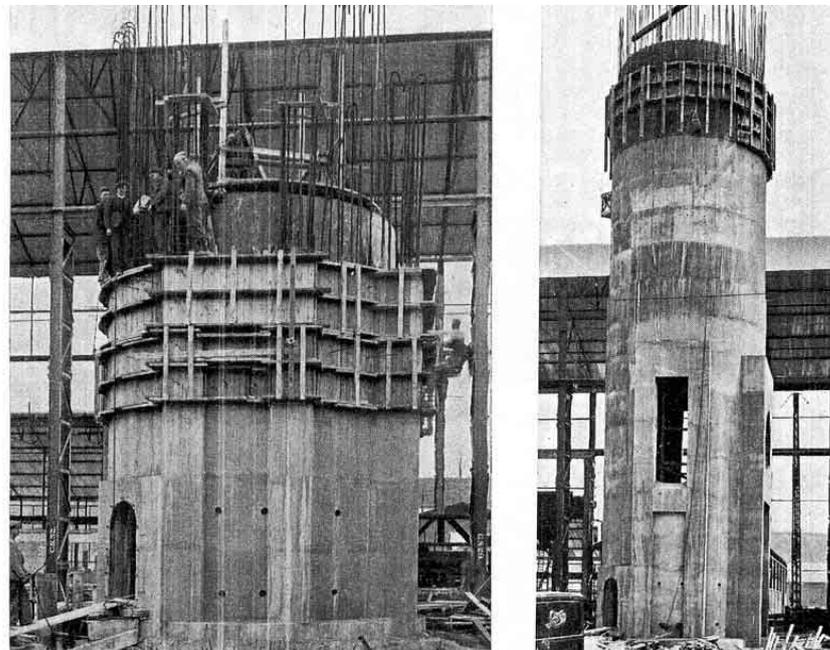


Figure 1 - Timber forms for a chimney with cross-sectional variations (Wynn & Manning 1926).

Concrete is the solid produced when cement, water, aggregates and admixtures are poured in to a form and left to cure. This change, from a plastic state which is somewhere between a liquid and a solid, makes concrete a very versatile building material. Varying the constituents or design of the mix can change the properties of the concrete in both its states, but also its transition between the two. If they are affordable, improving certain properties of concrete - such as workability or environmental impact - is desirable irrespective of the scenario for which the mix is designed. Exhaustive lists of mix designs are well documented and will not be repeated here.

Rapid setting concrete is most relevant to continuous or incremental casts, since these processes risk being too slow to be tenable. The curing process can be expedited by admixtures which speed up the hydration process (accelerators) and create better bonds between new and old concrete (bonding agents). The standard single pour is very rapid, but the entire cast must be described by the formwork (Fig.2).

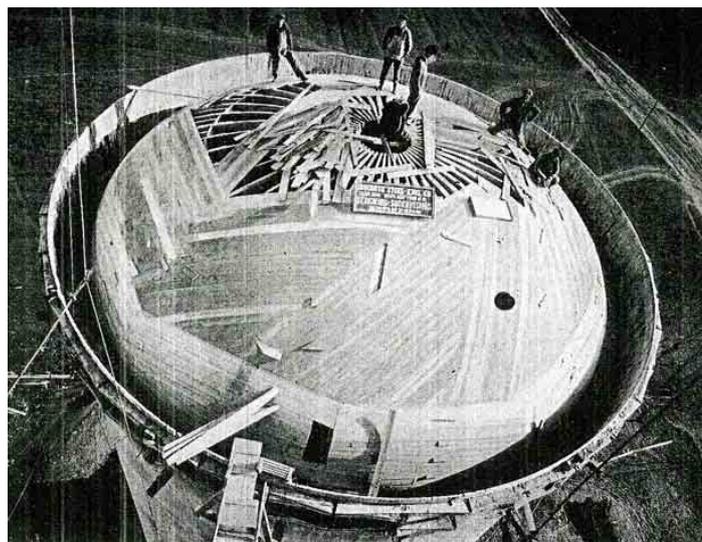


Figure 2 - Forms to describe a domed roof (Wynn and Manning 1926).

This is a very expensive and wasteful process.

For most structures, more time and cost are required to make, erect, and remove formwork than the time and cost to place the concrete or reinforcing steel. For some structures, the cost of formwork exceeds the cost of concrete and steel combined (Oberlender, 1996, p.1).

Requiring the concrete to set rapidly reduces the lateral pressure exerted by the concrete on the formwork. The pressure is dependent upon a maximum of five variables: the rate of placing, temperature, density, consolidation and depth of the concrete. This depth is also minimised by the slipforming process. This means that even if it proves slower when compared to a standard single pour, wastage is minimised while the risks of the frame deflecting or failing completely is greatly reduced. The logic that dictates the material choice for the frame of any formwork is much the same as the decision making process which determines any structural material.

Seeking to communicate the complexity of formwork, source material divides designs into very broad categories. These generalisations and discrete categories negate the possibility that adaptable formwork might be capable of negotiating the transition between footing and column. There is a presupposition that familiar building elements such as walls and roofs are distinct forms with disparate formworks. Slipforms can be adapted and have accordingly “been employed successfully and economically in situations which have required discontinuity of section” (Richardson, 1977, p.247). Parts can be constructed to slide over each other so that they form telescopes like a camera lens or be added or subtracted if it jumps. Experiments have been conducted to cast parabolic oil platforms, but the Expiatory Church of the Sagrada Familia is the most famous and sustained pursuit of new

construction methods to describe complex columns. Multiple generations of architects and engineers have sought to realise the intentions of the architect Antoni Gaudi. Sometimes they have adhered to fragments of surviving documentation, often they have needed to embellish his calculations with new constraints, but always they have searched for the simplest formulae to describe his designs.

Mark Burry has documented the recalculation of the nave by the structures department of the Polytechnic University of Barcelona (Fig. 3).

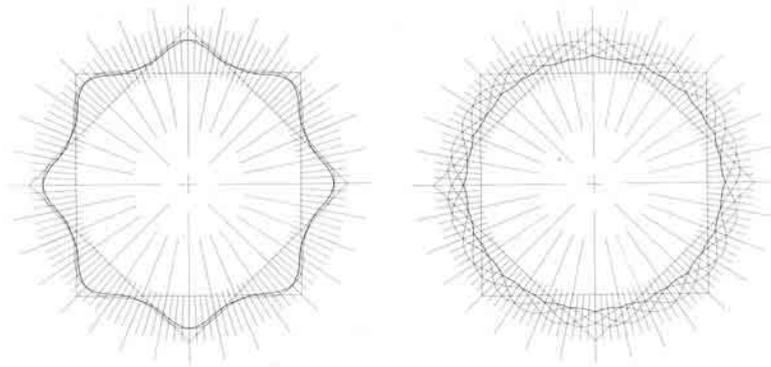


Figure 3 - Column Sections generated by the interference between two helicoids (Burry, 1993).

The columns are objects of great beauty and their apparent complexity disguises a simple formula for the generation of each type. Each starts with a base profile. This profile doubles on itself and acts helicoidally for each stage of the column but in opposite senses: one minutely rotates clockwise up the length of the column, the other anticlockwise. The column is that solid common to the two opposing helicoids, that is, the material left by the interference between the two (Burry, 1993).

Other columns located in the nave are hyperbolic paraboloids formed “from a surface of lines (the generatrices) being pulled straight between non-coplanar

borders (the directrices)” (Burry, 1993). These naturally occurring surfaces have even been employed to produce representations of entire buildings (Prousalidou, 2006). It seems possible that almost any imagined built form might be described as “the points of origin and termination of an appropriate number of straight lines” (Burry, 1993). Whether fabrication is possible from this description however is debatable. Pride and tourism have funded the unfinished Expiatory Church of the Sagrada Familia when other projects would have been abandoned. While the principals of construction evolved for this project might be transferable to others, it is unlikely that the financial investment, research and skill that they require will be too (Fig. 4).

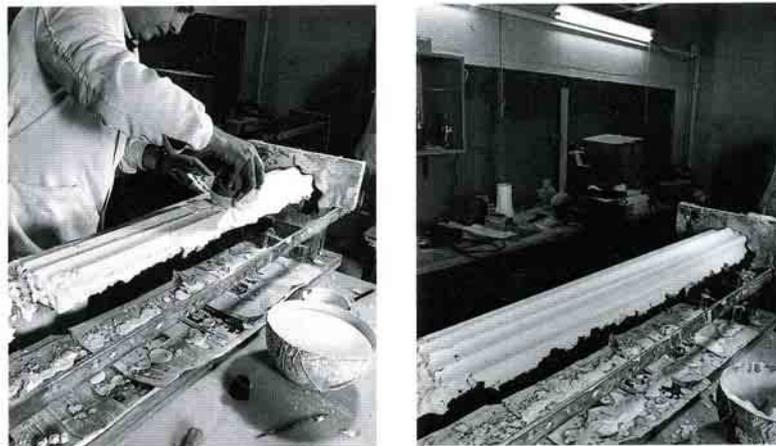


Figure 4 - The making of a column model based on a system invented by the model makers which allows the sequential counter-rotation of the zinc profile over the setting plaster of Paris (Burry, 1993).

Knowing that he would never see its completion, Gaudi began to develop “a codex based upon ruled surfaces in operation, providing for the first time, a ‘describable’ building, as opposed to one whose construction was based on sketches and day-to-day discussion with the operatives involved” (Burry, 2001). Skilled builders have constructed complex and hugely wasteful traditional formworks to cast the descriptions of these forms while craftsmen have retrospectively invented tools to painstakingly carve columns by hand.

Perhaps the machination of constructing a limited number of ruled surfaces should have been the focus of research before advocating their use to describe the entire structure. The design of the Expiatory Church of the Sagrada Familia could then have been restricted to the ruled surfaces most easily built by this new system. It is interesting to speculate whether this alternative chronology of events would have imposed perceivable or unperceivable constraints upon its complexity.

The implementation of new technologies to mechanise the design process from conception to fabrication continues to improve the ratio between the input of information, raw material, time and money and the return of design documentation, processed material and profit. At the same time, the personal fabrication revolution prophesised by Neil Gershenfeld (1995) depicts a future in which design will be democratised and meet demands more explicitly than ever before. Not only can alternative options be probed faster, more rigorously and more precisely by a computer, but these possibilities can be more precisely planned, predicted and fabricated by a machine too. Their most obvious limitation is the size of their 'envelope' which dictates the maximum dimensions of any part they fabricate.

In response, research has investigated the potential of self-assembling structural components (Sass 2004). If they can be embedded with mechanisms to dictate how they assemble upon contact, then the information, skill and time required to assemble a structure will be drastically reduced. This approach is analogous to creating bricks cut by a machine to aggregate in a certain bonding arrangement. Someone with no knowledge of brickwork could unknowingly lay an English or Flemish bond lacking any experience with mortar or skill with a trowel. Despite this economisation, since a self-assembling structure need not be rectilinear, nor rely purely upon the

compressive strength of its components, so the structural possibilities are greater. This is certainly a more efficient process, but one that evokes certain properties of emergence too.

If one, specific structure is sub-divided and then its components redesigned so that they can tessellate only with their neighbours, then this approach risks being similar to a jigsaw puzzle and inheriting the same search for the correct piece. An adaptable component with a manageable number of possible combinations with other components is less complex, especially if those which are assembled first dictate the forms adopted by subsequent components. No searching, careful labelling or layouts would be necessary and the resulting structure would be one of many possibilities created by a set of self-assembling, adapting components.

Aims and Objectives

This study digitally models the physical procedure of constructing multiple instances of a modular structural system intended to be appropriated for a process of incrementally casting concrete. The system can then be informed by a digital exploration of its possibilities and re-prototyped to build digitally conceived structures in reality.

An efficient system and an efficient process of design are integral to the thesis, but are not the primary contribution to knowledge. The physical objective is to reduce monetary and material wastage with the development of a scalable system which can adapt to incrementally cast complex concrete structures with cross-sectional discontinuity. Deceptively simple descriptions should dictate the forms produced by the system if the underlying geometry of the system is understood or pre-programmed in the computer. The number of these controlling parameters should be minimised to ensure economy of forms. Although capable of complexity, it must be easily assembled by unskilled operators to be tenable.

Digitally it is aimed to show that if the method of assembling such a system in reality is replicated by the digital model, then it is not always necessary to distinguish between directional (unequal) and bi-directional (equal) relationships and model them differently.

Once we allow bi-directional relationships between components, the cognitive complexity explodes. Even with a reasonably-complex directed graph model, we are challenging the cognitive limits. So in the main, bi-directionality is not essential and is difficult to deal with.

There are, however, important aspects of design where design intent cannot be modeled as simple dependency relations. Therefore, bi-directionality is an important requirement that needs to be addressed.

[Using a constraint solver] requires a different approach to defining components and relationship used in the previous graph dependency approach (Aish, 2005, p.8).

This thesis attempts to show that if an exact process of construction is modelled in GenerativeComponents (graph-dependent, associative, object-orientated and parametric modelling software), simple dependency relationships established between extra geometries can permit them to act as constraint solvers.

If solving constraints is the bottom-up approach to the problem of modelling the prototype, then GenerativeComponents is the top-down alternative. Dimensions do not have to be specified until all of the relationships between components have been defined at the end of the modelling process. In order to explore different permutations, the design model does not have to be rebuilt. This distinction is important and worth reemphasising to introduce the correct terminology. The geometry of a new object (the prototype) must first be modelled from the existing library of components, each of which employs different techniques to recalculate location, size and appearance. The properties of a component, which either control or describe its current state, can be added subsequently to observe their impact.

Once all of the components and inter-component relationships which generate each instance of the prototype for this application have been established, they can be converted into an object or 'feature' itself and

multiple instances created and assembled if its inputs and outputs are correctly defined. The object-orientated approach simplifies the reuse of this new feature

The first instance of it dictates the location of the structure since all subsequent placements must be upon this first object and deviate no more from the axis defined by this initial placement than is structurally possible (unless the axis is curved and the growing structure supported by a temporary structure). Changes to the location or geometry of the base object, must propagate through all the objects it supports. Each new instance of the prototype must consequently refer to the geometry of the 'upstream' instance held in place by the set concrete, but will also inform the shape of subsequent 'downstream' instances waiting for concrete to be poured in to the increment of the column they describe. How this occurs, depends upon how the prototype is constructed and the update methods it takes as a user generated feature.

It is important to emphasise that changing the geometry of any instance of the prototype will only impact upon subsequent instances. A constraint solver, in comparison, would adjust all instances in response to any change, potentially producing an unviable structure which fails to acknowledge the chronology enforced by incrementally assembling the formwork or any other structure.

While a change in the riser height might affect the number of risers in a particular stair flight, it is unlikely that a designer would want to jack up the whole building when increasing the riser height (Aish, 2005).

GenerativeComponents justifies its graph dependency approach on this basis. It is unlikely that a riser, which is an obviously subsidiary structure, will dictate the distance between floors. This seems a very convenient example in the context of more complex buildings and structures, but even a simple example describing very traditional building methods exposes the limitations of this argument which are acknowledged in the written material released with the beta version of the software.

If a brick is considered instead of a riser, then the hierarchy is less obvious. It becomes less clear which property is always more important in this scenario: the number of bricks or the height of the floor which they support. Since the brick has a structural function and supports the floor, the two are neither independent, nor is one universally more important than the other. GenerativeComponents does not permit this equivalence and forces the user to establish a preference.

To test this reasoning, a structural system speculatively designed to incrementally cast concrete was prototyped. This one component was then digitally modelled in GenerativeComponents - graph dependent, associative and parametric modelling software - then cross compared with and tested against the behaviour of the physical prototype. Once a faithful correlation had been achieved, different tessellations and aggregations of the prototype were explored digitally and the system embellished. This then fed back and informed the design and manufacture of a second physical prototype with two components. This refined system will be tested by casting a concrete column, completing a complete loop of physical and digital modelling with the fabrication of a real building component.

Methodology

With the physical prototype inextricably linked to the development of the digital model, it was important that the emphasis of this study remained upon its digitisation rather than extensive testing of materials and mechanisms. This substantiated the initial decision to pursue a jumpform. Other speculative methods utilising a flexible sheath capable of continuous casting were postponed for further study. As with any controlled experiment it is important to only change one variable and first digitise a simpler, more predictable system: the more familiar and tested process of repeatedly releasing, stripping and replacing the form was judged less likely to fail.

Physical Prototype One

Form

Before digitisation could begin it was necessary to assert; the number and shapes of the faces, the supporting substructure and the hinges determining how the prototype moved between instances. In addition to these basic prerequisites, the search was limited by the stipulation that the prototype be equally capable of casting square and more subversive sections while multiple prototypes should be capable of interlocking (and suggest that, with some refinement, they might self assemble). These deliberately vague criteria are intended to encourage and permit feedback from the process of digitisation. The first prototype might be enhanced by details highlighted by the digital model or if foresight is particularly poor, be abandoned completely. Separating the frame from the faces seemed an extension of the logic that the prototype would not be informed by extensive physical testing and should be adaptable in the event of material choices failing or adhering to the concrete. Initial adaptable frame tests, built from augmented Lego, had

the advantage of being inherently parametric. These tests began the paradoxical pursuit of an adaptable frame which, although predictable in isolation, when applied to itself would have the potential to produce structures much harder to envisage - without a computer.

Frame

Although intuitively appropriate, working with Lego was problematic. The perceived advantage of this construction toy to assemble in discrete aggregations meant designing specific solutions with non-specific components. The increasing complexity of the simplest conceivable mechanisms designed to angle the faces of the formwork seemed incompatible with a scalable system. The weight of the frame would have had to increase rapidly in relation to the size of structure it was capable of casting and this seemed precarious with the previous casts supposed to support the weight of the form while subsequent increments cure. Underlying all of these problems was the crux of the design challenge: immobilising an adaptable formwork intrinsically conceived to be repositioned and assume different states.

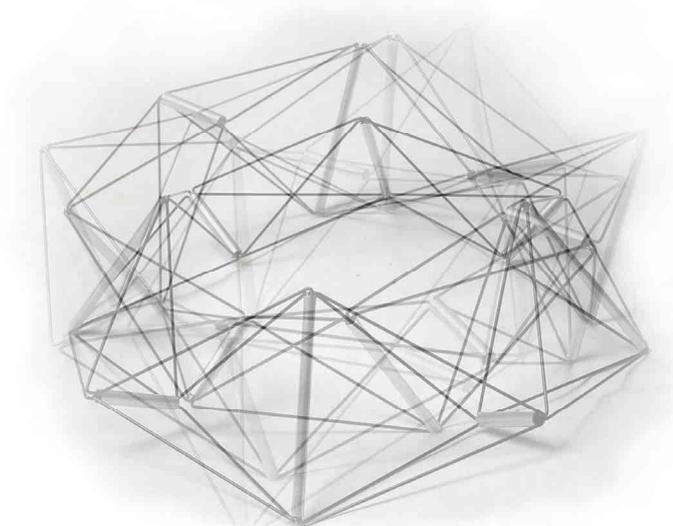


Figure 5 - Multiple states of Prototype One.

The Lego experiments refined the search for alternative construction systems capable of continuous assemblies and inherently more appropriate. A simpler alternative construction system was found without the issues foreseen with the Lego experiments. This unbranded set, consisting of two different parts, was assembled to create a triangular structure which was strong while static, but also capable of complex dynamism (Fig. 5). The behaviour of its repetitive and elementary geometries seemed simple to predict, but it was much harder to foresee all the different and finite ways it might aggregate (Fig. 6).

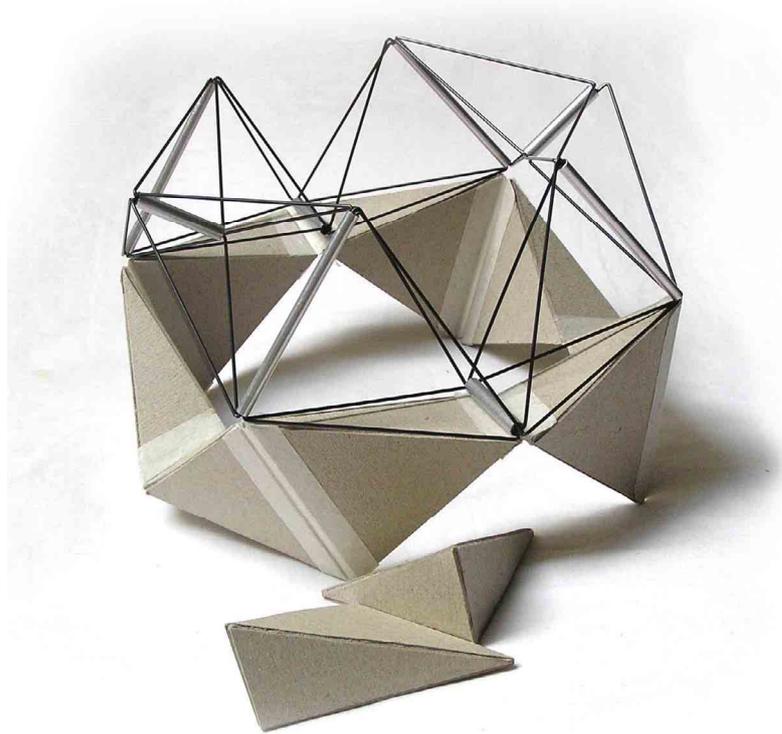


Figure 6 - Two stacked instances of the physical prototype.

Faces

Before the possible aggregations could be fully tested by the digital model, only general conclusions were possible about the material from which the

faces that touch the curing concrete should be cut, as well as the release agent which would coat them. Plywood, solid timber and steel are commonly used, but the use of aluminium, plastic, magnesium, plaster of Paris and fibreglass is occasionally appropriate too. Faces must be adequately strong and rigid, sufficiently smooth and economical (considering initial cost and number of reuses). A combination of plywood faced with aluminium or fibreglass is particularly advantageous since with careful handling the latter combination has been documented to yield one hundred uses before maintenance is needed (p.21 McAdam & Lee 1997). Fibreglass is also structurally adequate to be used in isolation or to reinforce a vacuum moulded thermoplastic sheet such as PVC (p.22 McAdam & Lee 1997).

Reusing any of these materials, the choice of release agents is important to prevent scaling (weak concrete sticking to the stronger form) or scabbing (weak formface sticking to the concrete), but to also seal off the formface and prevent moisture absorption which might damage it or completely prevent reuse. Neat oils and mould cream emulsions are generally not recommended. All of these possibilities had to be considered prior to a working digital model of the system as material choice was dependent upon method of attachment which was in turn dependent upon the mechanism employed to lock together each instance.

Standard Components

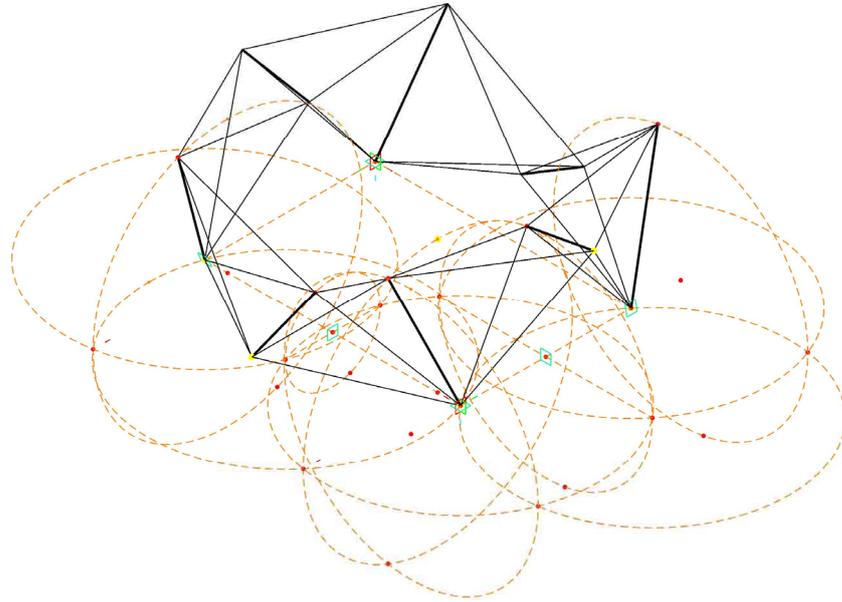


Figure 7 - One instance with visible construction geometry.

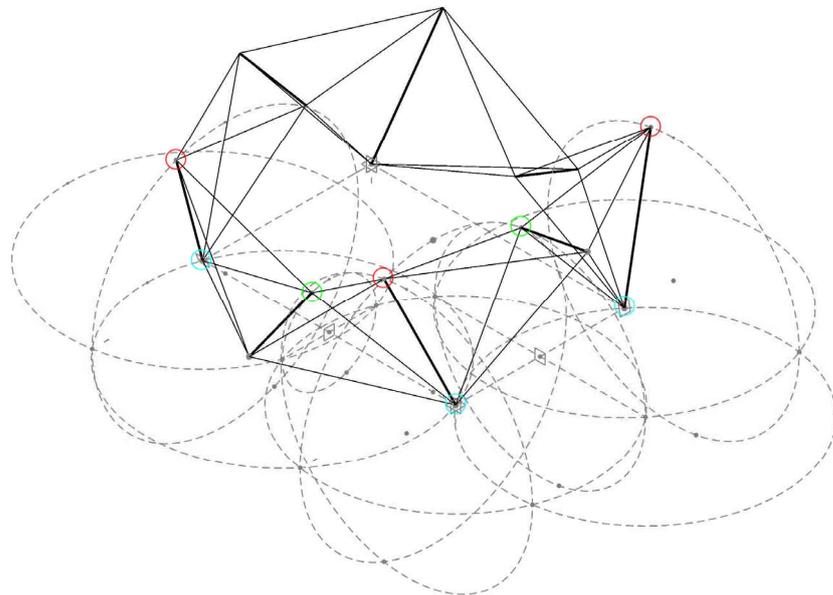


Figure 8 - Primary points (blue), secondary points (red) and meeting points (green).

Primary Points

Observing the behaviour of the physical structure hinted at the possibility that the location of just three primary points (Fig. 8), if their Z (vertical) depth is

fixed, is sufficient to calculate the entire structure. This is evident when three points are pinned in place by hand on a flat, horizontal surface in reality and the frame cannot be contorted beyond the movement granted by the tolerances of its hinges. Opposite faces are identical too, so the intermediary structure between these three primary points should be replicable to complete any instance. This repetition means that only two of the four internal planes must be constructed.

In theory, every component should have a direct relationship with just these three primary points or an indirect relationship with them via their dependents. The trajectory of the column cast by the system is always tangential to the horizontal levelling of the concrete under gravity, so it is tempting to refer to the base coordinate system to create a horizontal plane which bisects the three primary points. This is not good practice. It is better to refer to the primaries and specify the origin, X and Y points of this new plane instead. It is restrictive to assume that it will always be possible to reference the base or any other coordinate system. Internal references to other components (which will eventually be assimilated to create a new feature) are therefore preferable as it is difficult to foresee all the eventual applications of any prototype.

Possible Meeting Points

This horizontal plane supports the triangulation of two faces in two-dimensions before they are rotated to meet in three-dimensions and copied to complete the construction. These faces of the prototype are those with the potential to be completely or partially translated into the surfaces of a significant depth of concrete. Created by three or four (two equilateral and one or two isosceles) triangles, they can be simply defined as those which will make contact with the curing concrete and be translated by the system

into the faceted surface of the finished cast. Whether the triangles which constitute each internal plane number three or four in any instance, they all have one common meeting point (Fig. 8).

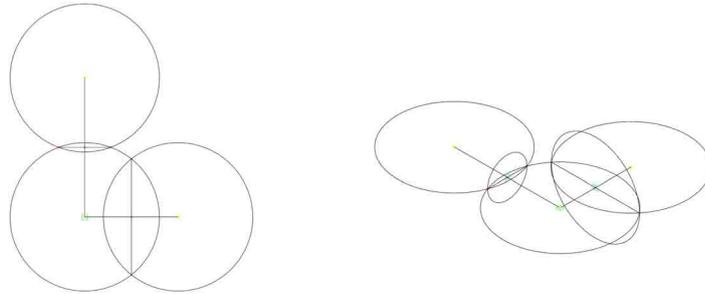


Figure 9 - Possible locations of meeting points created by the intersection of two spheres.

Four possible locations of the two real meeting points can be found by creating three circles on the horizontal plane (Fig. 9), updated by centre point (one of the primaries) and radius (the structural member size). These two pairs of points lie on, but are also, the diameters of two more circles which, created at the middle point of two lines between the primaries and at a tangent to the horizontal plane, depict every hypothetical location of the two real meeting points.

Possible Secondary Points and Possible Planes

Which of these possible points is the real meeting point in any instance depends solely upon the location of the three primary points, but is not directly calculable from them. From two of the primary points, the three or four triangles that together define one of the faces can be generated in two-dimensions. Where this plane is precisely located in three-dimensional space, however, is only determinable when the shape of both faces is known. The technique is to construct the triangles using intersecting circles with

member length radii on a hypothetical plane that might be interpreted as a temporary support or a ‘guess’ at the angle of the real one.

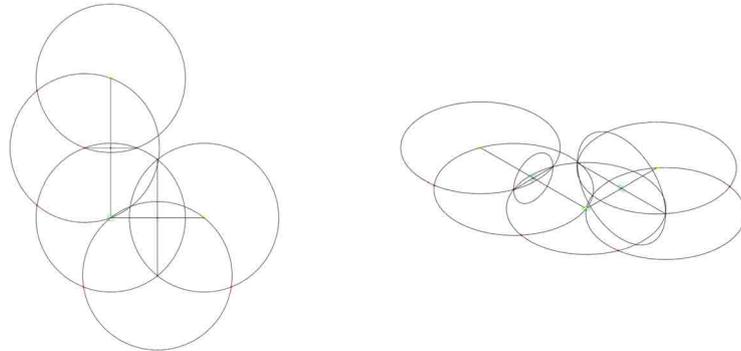


Figure 10 - Four possible secondary points created by five intersecting circles.

The horizontal plane is an obvious choice given that it already supports three of the five circles necessary to triangulate the possible secondary points. Two more circles created on it with possible meeting points as centres, intersect two of the existing three twice (Fig. 10), so it is necessary to embed the design with the appropriate conditionals to validate one point of intersection. Again this could be a simple comparison of X or Y coordinates between the possible meeting and secondary points, but as has already been discussed this reference to the base coordinate system is potentially problematic. It is preferable to build conditionals that refer to the lengths of invisible lines or distances from other components.

Real Points and Real Planes

Once the possible points of intersection have been selected, the geometry to establish the real secondary points can be constructed. The three primary points enable calculation of two two-dimensional projections of the members which connect them to the secondary points. These rotated projections are on the horizontal plane which bisects the primary points and must be rotated

back towards the tangent to this surface until it is discovered that, in combination, they describe a line in three-dimensions.

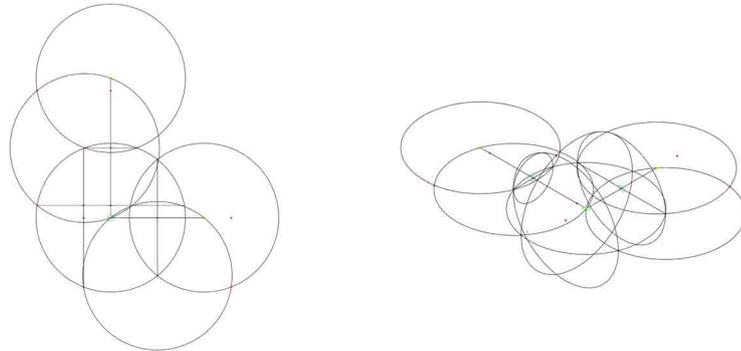


Figure 11 - The creation of a real secondary point by two intersecting circles.

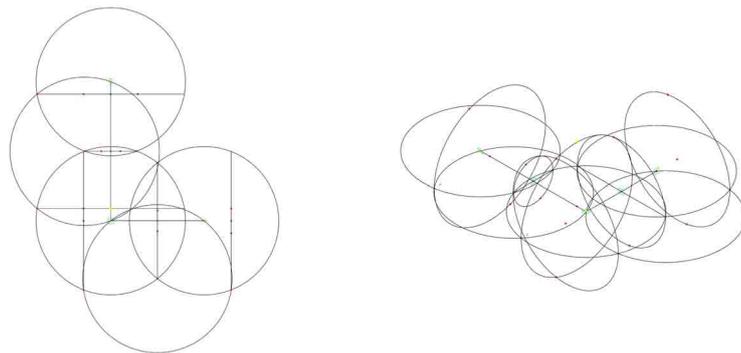


Figure 12 - The creation of the other secondary points.

This verbal description hints at the implemented method of construction: two circles are drawn from centre points which are projections of the hypothetical secondary points on to direction components (essentially infinite extensions of the bottom line of each face). If their radii are updated according to the length of the line between these points, then the point at which they intersect will be a real secondary point (Fig. 11). From this crucial component, both faces can be constructed in three-dimensions (Fig. 12).

The incline of the two internal faces can be created according to the location of this single point. The remaining secondary points are then created where

these planes intersect circles which represent all of their possible locations. The infinite nature of these planes means that more conditional scripting is necessary to ascertain which of the two points created on each circle should be referenced by future components. These points can then be joined appropriately to create the basic geometry of any instance and embellished with some extraneous components to improve the resemblance of the output to the actual prototype (Fig. 13).

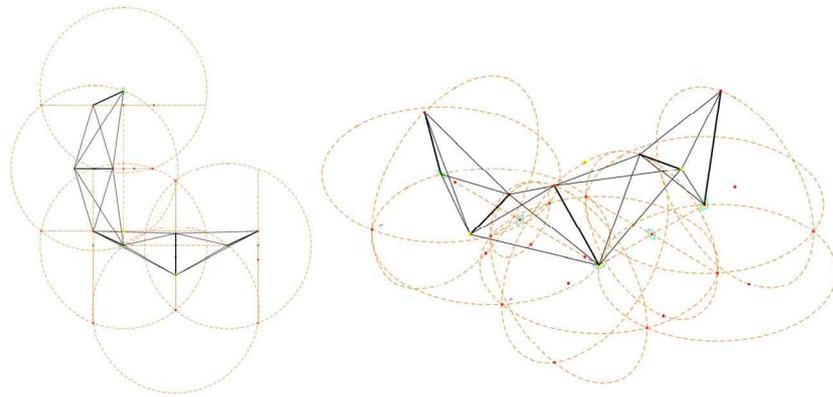


Figure 13 - Completed construction of one instance (subject to rotation).

Scripted Components

Documentation of GenerativeComponents remains concise. It was not always possible to construct the neatest or most obvious relationships between components. Lateral thought and recoding were frequently necessary to find an alternative to a more direct method. It was not always clear whether the digital model failed to mimic the behaviour of the physical prototype due (in order of their perceived probability) to semantic errors in the programming, flawed logic dictating relationships between components or limitations of the software itself.

When neither rigorous inspection of the code nor experimentation with alternative components and relationships fixed the failure of the digital model, it was deduced by process of elimination that the software must be accountable. The experiments to assert this were time consuming and warrant documentation when the problems and techniques, if not the circuitous specific solutions, are generally applicable. Specific applications of GenerativeComponents will obviously demand very specific code, but when an in-built solution could not be found, some of these crude circumnavigations advocate the refinement of existing features and development of new update methods.

Multiple intersections between features meant numerous transactions had to be embedded with the intelligence to discern between points. If multiple instances fulfil the criteria of a newly created component, then GenerativeComponents automatically creates an array to label and differentiate between them. It was obviously important not to code long lists of specific scenarios, but to use a universal comparison between multiple instances of the same point component, thereby negating the need to know every scenario and which point to specify should be referenced by dependent

components in each one. When the code had to choose between point A and point B for example, then this is best expressed in words as always picking which of these is furthest from a suitably constant point X.

Initially all components dependent upon these conditional points had to be created within a script transaction so that they referenced the correct point(s) and since these points were frequently part of multi-dimensional arrays (some with potentially problematic phantom dimensions), the length of the code increased while its comprehensibility decreased. It seemed that components dependent upon conditional components unavoidably inherit this undesirable property from their parents. Fortunately, there is a simple, but inelegant method to eradicate the incorrect points and references to lengthy arrays with a single conditional statement.

The principal is to code a script transaction with a universal conditional that compares multiple instances of a point to find which is desirable and then replace it with a new point created at the same location. This increases the number of the components, but the single manifestation of the new point can be referred to by all of its dependents. This reduces the potential for human error and probability of referencing phantom dimensions while permitting greater scrutiny of the points selected by the code. If GenerativeComponents always creates an array of points, this is a sustainable solution.

Certain parameters may percolate through a model to sometimes produce single instances, sometimes multiple instances and sometimes none at all. In which case, these rules must be broken. The programmer must identify the conditions, typically by referring to other components, which dictate all of the relevant scenarios. Sometimes several conditional statements must be used just to identify a single correct point. If the code had to again choose

between point A and point B, then in words the code would assess whether the distance between point X and point Y is greater than Z and pick point A or point B depending upon the result.

There is a danger that it might be impossible to identify and correctly code for every scenario. Fortunately digitising a physical prototype, the thresholds governing point selection were apparently identifiable. If examination and comparison of the physical and digital models validates that the code is selecting the correct point, then it can again be replaced by a new point. In order to avoid referencing the base coordinate system, the new point should be created not from the Cartesian coordinates of the point it is replacing, but by appropriating the method 'by centroid of set' and referring to the same point twice. This technique still permitted dependent components to reference a single instance of a constant point, but choosing between multiple instances of the same point created by the intersection of multiple features must be simpler.

Ironically, with so much code devoted to ignoring when Generative Components found too many intersections of components, the software frequently defied an expectation that two features would meet to produce multiple points of intersection. It is hard to be completely certain why, but the best hypothesis is that these inconsistencies were caused by rounding errors. These plague all computation and occur when a program approximates a large or infinite number of digits after the period (or comma) by truncating them and rounding the last digit up or down. This may permit faster calculation within a finite period of time, but the result is obviously different from that which would be obtained using the real, exact numbers. If this approximation is used in subsequent calculations of dependent

components then an accumulation of errors may result. There are several crude fixes:

- increase the scale of the model;
- give supposedly intersecting curves 'thickness' (only graphically possible in GenerativeComponents);
- create the anticipated point by constructing its location from another instance of it.

User Generated Features

Designing components which will combine according to a set of clearly prescribed rules is something which becomes more intuitive with use of the software. Many decisions were made subconsciously, but what these decisions reflected was that the approach adopted by this study resonated with the process of modelling in Generative Components.

An understanding of the whole was important, but so were the four 'stages' which actually have to be simultaneously considered when developing a system. A diagrammatic dissection of these levels can be supplemented with these considerations located between the lower and higher "levels of granularity" (Aish 2005) which inform them (Fig. 14).

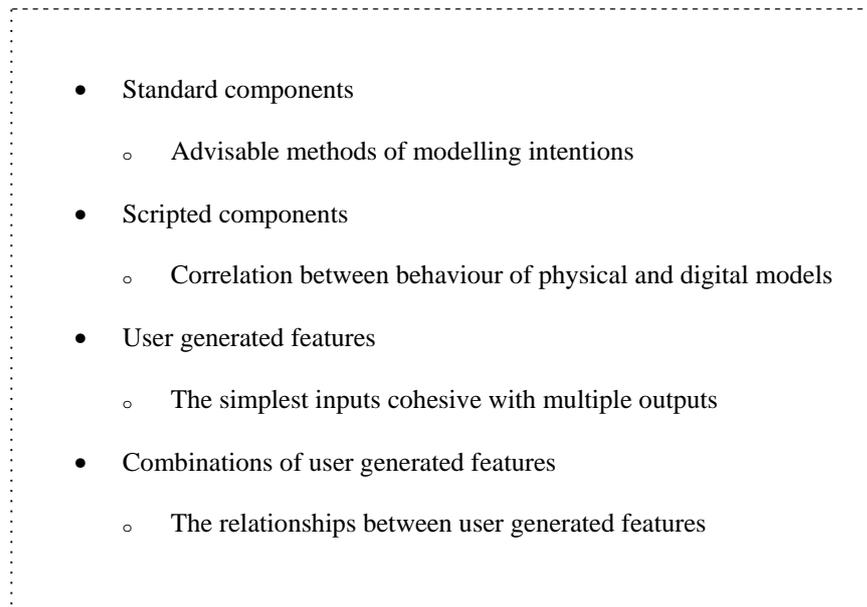
- 
- Standard components
 - Advisable methods of modelling intentions
 - Scripted components
 - Correlation between behaviour of physical and digital models
 - User generated features
 - The simplest inputs cohesive with multiple outputs
 - Combinations of user generated features
 - The relationships between user generated features

Figure 14 - The "levels of granularity" within the overall design of the system and the tensions between them.

The order of these levels relates to the chronology of their development, but that is an oversimplification of the real process which occasionally demanded

small loops to refine flaws in the logic of previous stages while ultimately navigating between the two fixes: the standard components and the design of the overall system. This is possible because the capabilities of GenerativeComponents are not fixed, but continually evolving.

Unlike more traditional CAD software, it permits the user to see ‘under the hood’ and adapt components or create completely new ones. More complicated problems involve coding script based transactions when the standard palette of model based transactions does not suffice. This hybridised approach does not preclude novices, unfamiliar in the C# language, from building more conventional structures nor expert programmers wishing to augment the software with components of their own creation. In fact, the latter may wish to compress and hide large dependency graphs for use by the former by creating a user defined component. Like the prototype it was employed to model, GenerativeComponents is a framework and all of its potential applications remain unknown.

The process of creating a feature is simple if how it will combine with other components has been considered during its creation. As the prototype is generated from just three points and these create two sets of three points (three meeting or three secondary points) which could locate a second prototype, the inputs and most crucial outputs of the new component were easily identifiable. The process of creating it presents the option of selecting whether features upon which these point outputs are construction (visibility can be turned on and off) or internal (invisible) as well as the chance to include dependents. This enabled components created for the purpose of visualising the output of the system to be included, in addition to the primary and secondary points capable of describing any permutations of the system on their own.

Combinations of User Generated Features

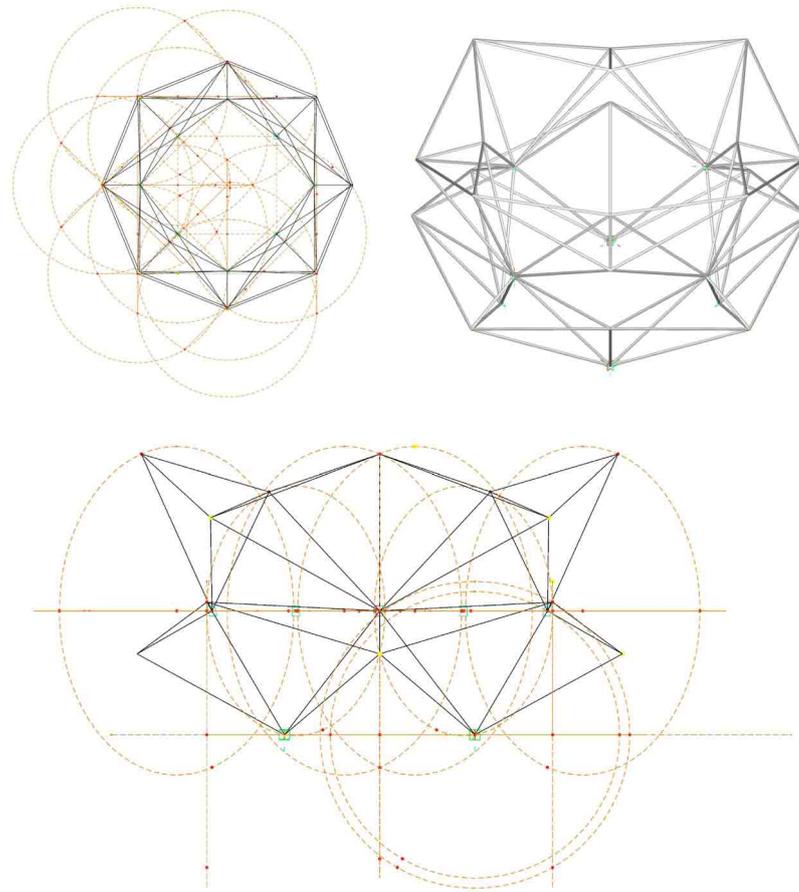


Figure 15 - Two instances of the system combined 'by meeting points'.

The system was digitally combined in two ways. The three primary points can be located on three of the meeting (Fig. 15) or secondary points (Fig. 16) to produce two different responses to any shape assumed by the previous component. Every permutation requires a different interstitial shape to fill the gaps created, but the method of attaching instances must vary too if both combinations are to be permitted by the system (Fig. 17). Supported between instances of the prototype it was noticed that hinging the interstitial faces might expand the possibilities of the system, if these non-structural parts could be offset by a single parameter from the most obvious points of connection along the curve representing the substructure of the prototype. This would require the same number of unique pieces.

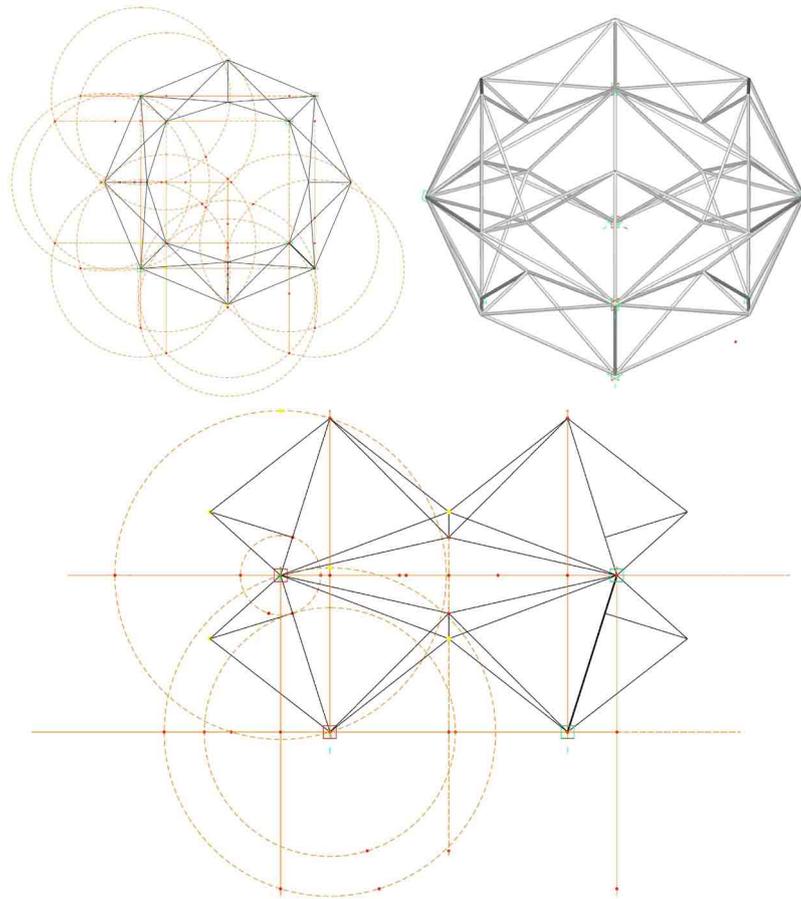


Figure 16 - Two instances of the system combined 'by secondary points'.

They might be perceived as more difficult to calculate, but GenerativeComponents has inbuilt functions to template polygons and organize them to be cut manually or by machine. Informed by expectations that form faces can be reused over one hundred times, it seemed acceptable to disregard the necessity for many more faces of the prototype to contact the curing concrete. It seemed less permissible to under exploit its versatility if the actual mechanics of these embellishments could be resolved in reality.

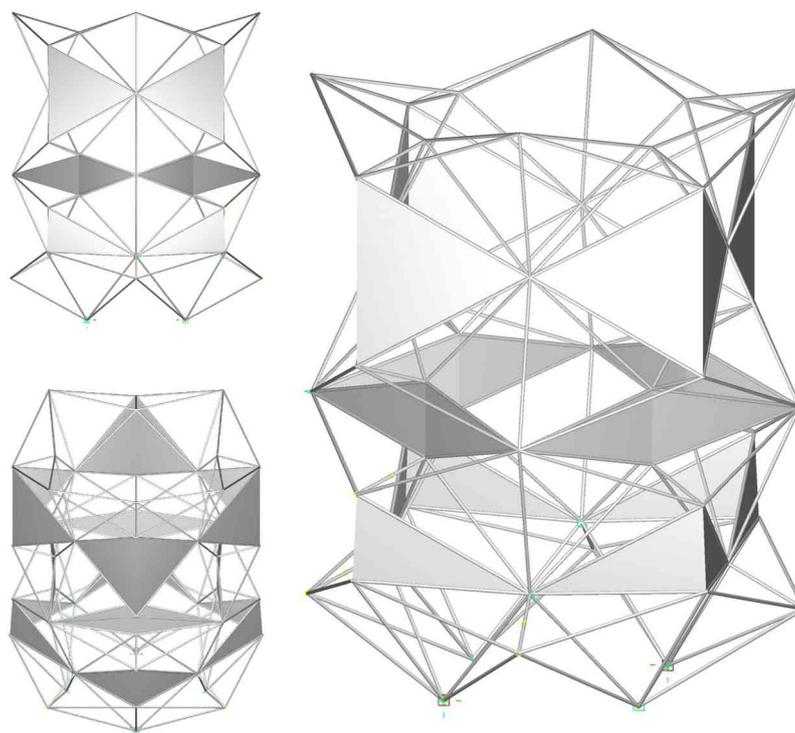


Figure 17 - Four instances of the system combined 'by meeting points' and 'by secondary points' to show the three different types of interstitial parts.

Results

The results depict the parametric properties of the model by means of static images exported directly from GenerativeComponents. The first set presents a cross-section of the infinite possibilities produced when the system is combined 'by secondary points'. Specifying the values of two distances (X & Y) between the three points which locate the first instance of the system is sufficient to describe one possibility. Both of these values have the same range as a consequence of the symmetry of the system. From this, all combinations of the extremes and middle values of these two ranges were built minus repetitive forms (Fig. 18). Rotations of identical forms were deemed unnecessary.

X	Y	X	Y	X	Y
11	11	11	15	11	19
15	11	15	15	15	19
19	11	19	15	15	19

Figure 18 - Table of eliminated and generated forms as simple two parameter descriptions.

This process of elimination produced six different frames, each of them documented by three drawings: two elevations and a plan. This was supplemented with three renders of each permutation. Two renders show two possible casts produced by the system with all interstitial parts set at the two possible extremes and one of the frame which could produce them (Figures 19-24). Each permutation has an infinite subset of iterations, if the value(s) of the parameter(s) dictating the interstitial parts is altered.

It should be emphasised that the drawings have been proportionally scaled to produce structures of equal height. This creates the illusion that the members change in length to achieve this uniformity.

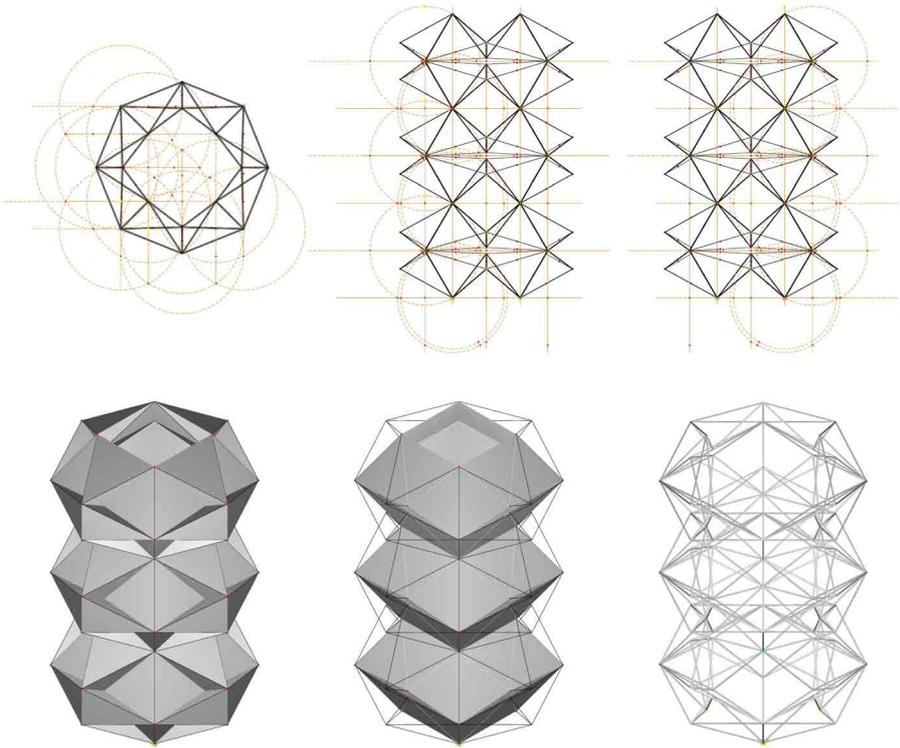


Figure 19 - X11Y11 drawings and renders.

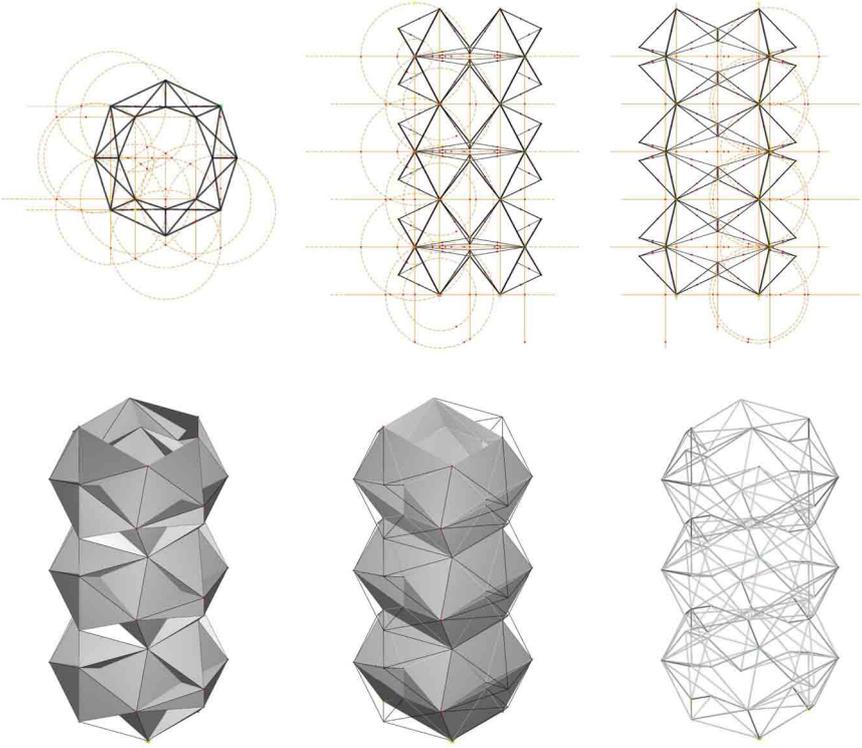


Figure 20 - X11Y15 drawings and renders.

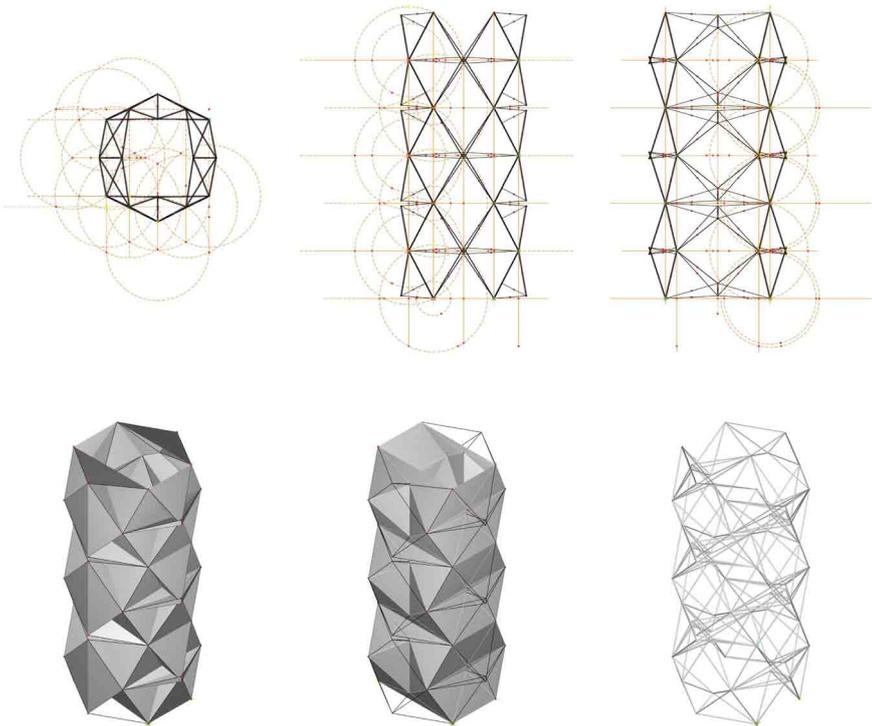


Figure 21 - X11Y19 drawings and renders.

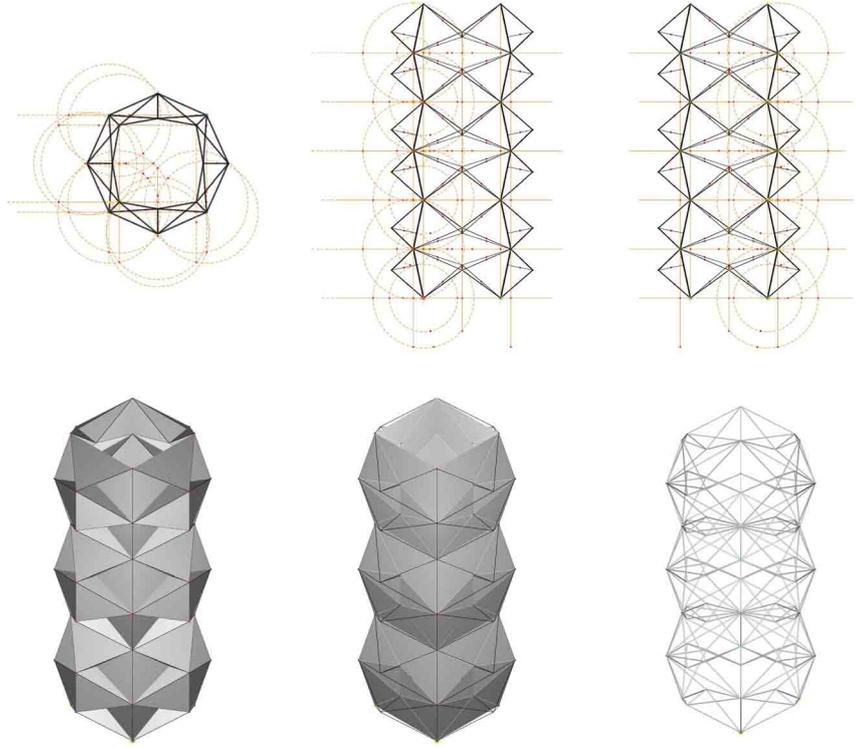


Figure 22 - X15Y15 drawings and renders.

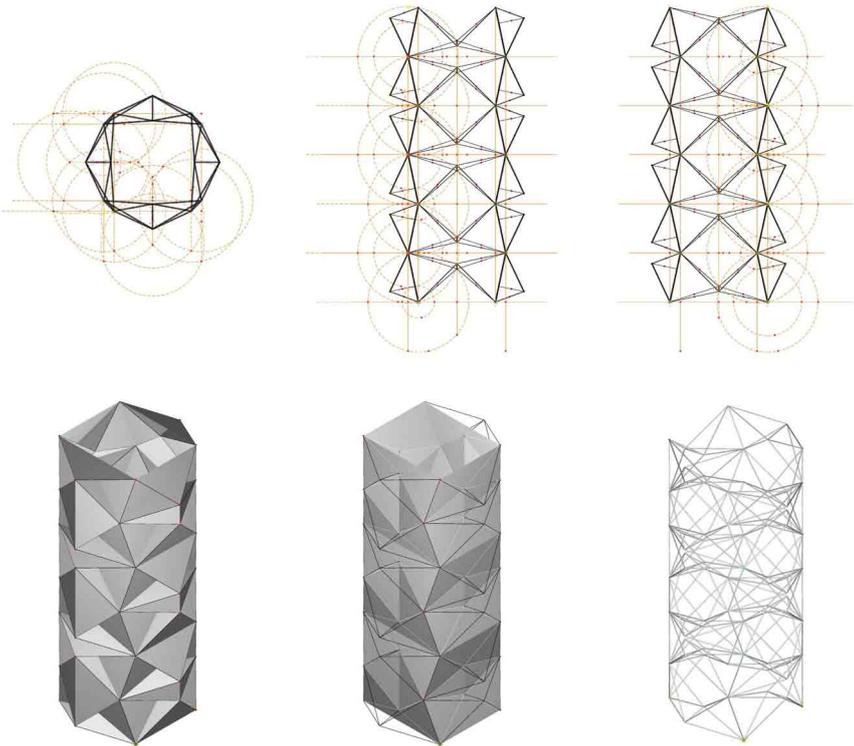


Figure 23 - X15Y19 drawings and renders.

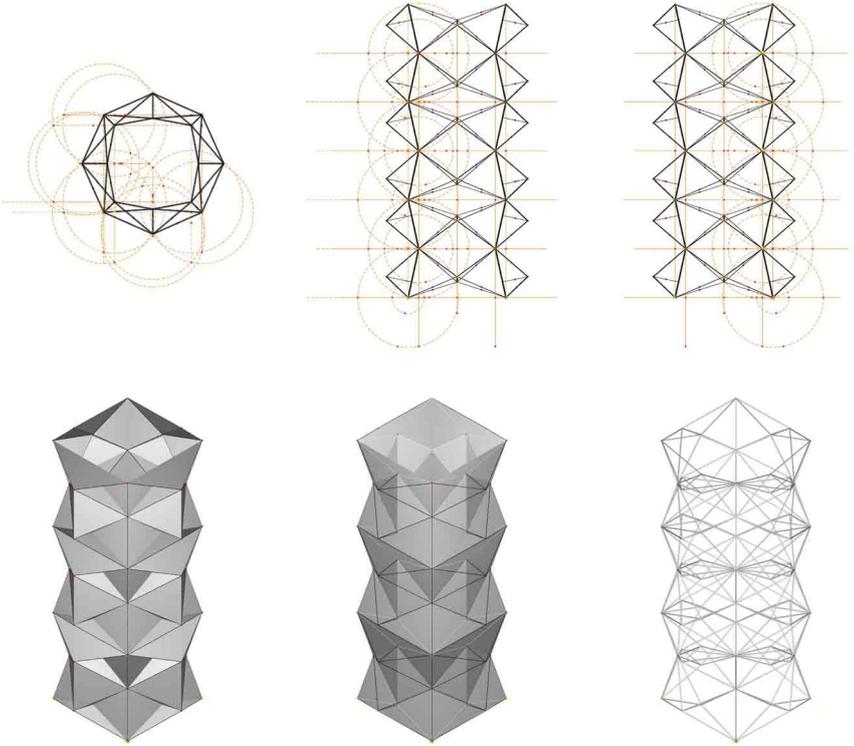


Figure 24 – X19Y19 drawings and renders.

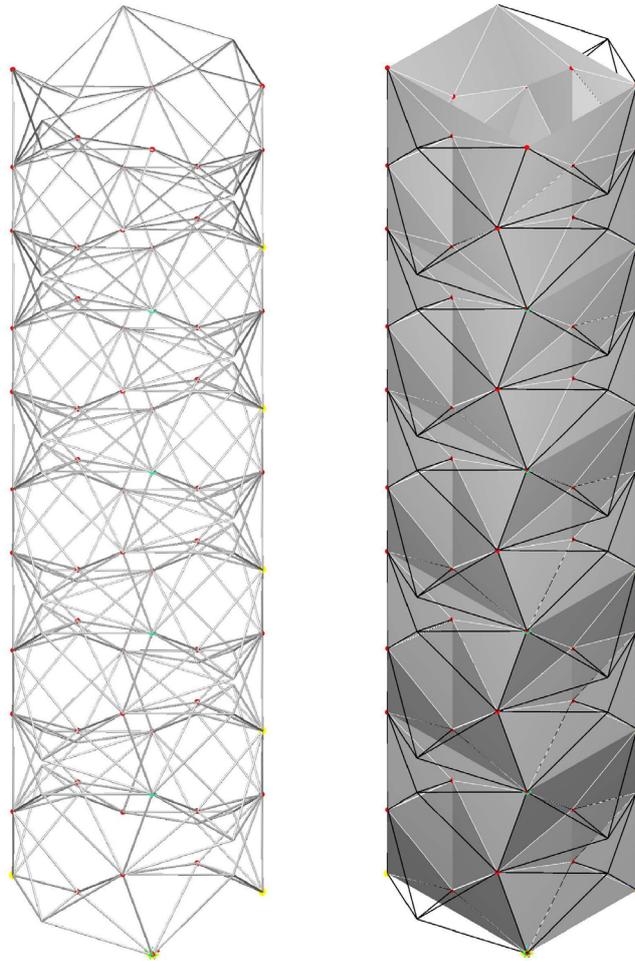


Figure 25 - Ten instances of the system generated from X15Y19F0.

The second set of results introduced a third parameter which was applied to the system to offset a new point from the secondary points of even instances. These new points were selected as the input for odd instances instead of the secondary points from which they were offset. The impact of this free parameter (F) was explored with ten instances of the structure produced by just one pair of distances (Fig. 25). The range of offsets was established and then the minimum, middle and maximum values were applied to this one permutation. Elevations (Figures 20, 21 & 22) and renders (Fig. 23) exported from GenerativeComponents documented the results.

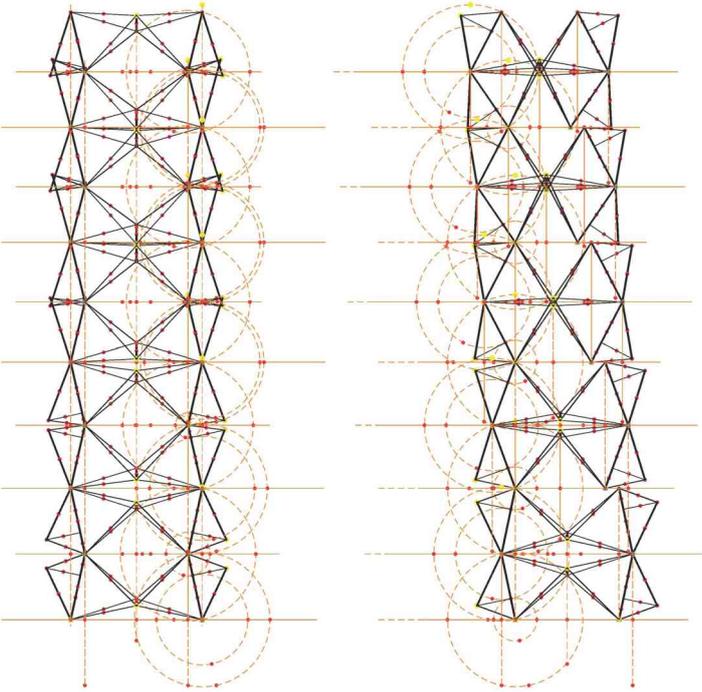


Figure 20 – X15Y19F2 elevations.

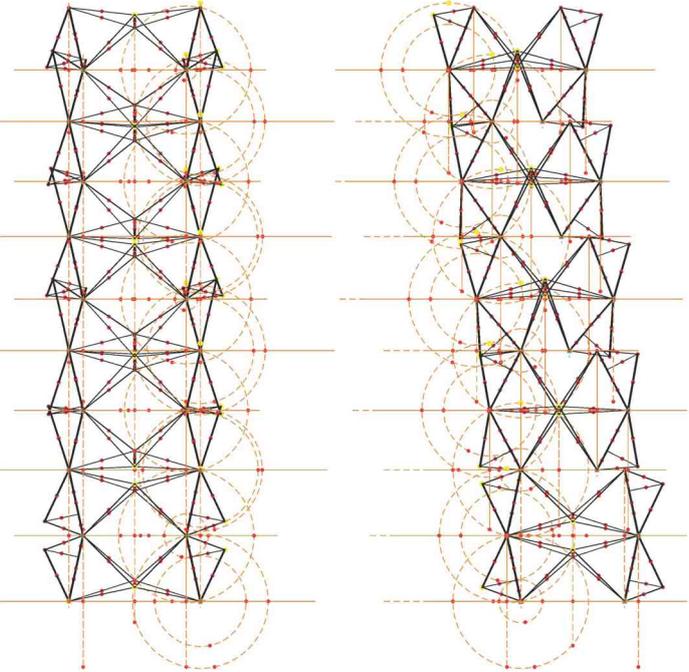


Figure 21 – X15Y19F4 elevations.

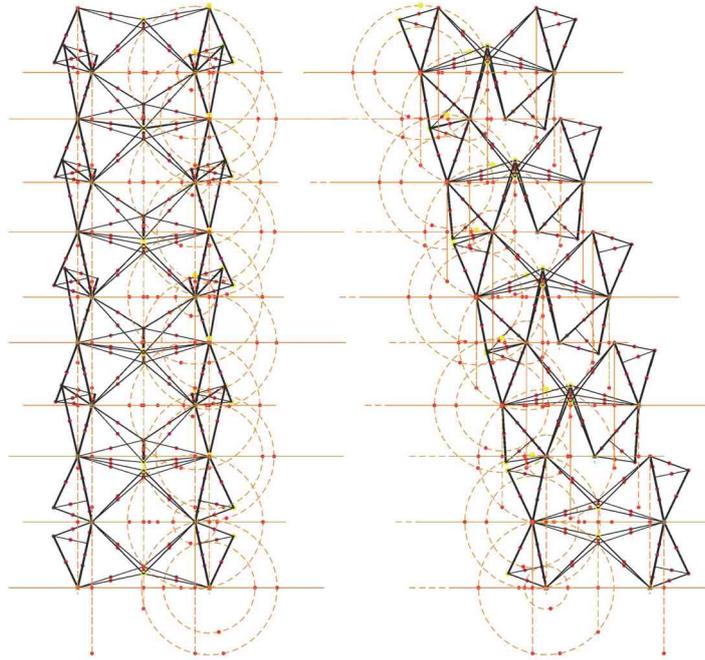


Figure 22 - X15Y19F8 elevations.

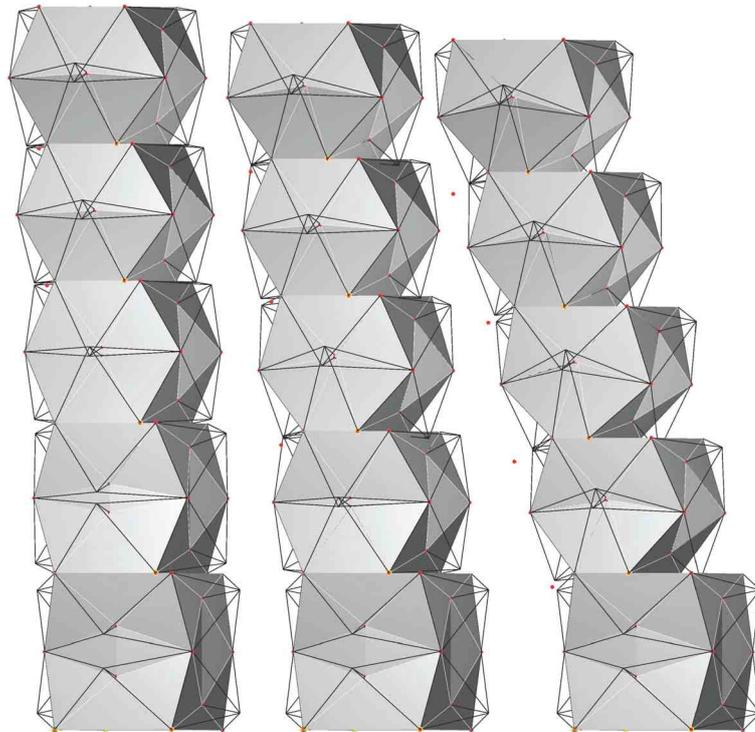


Figure 23 - X15Y19F0, X15Y19F0 and X15Y19F0 renders.

Discussion

Digital Characteristics of the System

The graph-based dependency model exactly replicates the real, physical process of aggregating the prototype. In the simplest scenario, the system exhibits two degrees of freedom. Two variables specify the distance between three points and the first instance is constructed from these. Then the primary points of each new instance are placed upon the secondary points of the last. Just two values control the output of this parametric system. A change to either propagates through the whole system and every instance is regenerated. This reflects the reality that once the first instance has been filled with concrete, these two parameters have been set. The results demonstrate that despite their repetition the casts of the space described by the system are hard to visualise from all instances stacked as if awaiting a single pour. It is also demonstrated even harder to extrapolate from just the values of the two parameters what this minimal amount of information will produce despite the repetitive nature of the system.

Other variations are made possible if new points are specified by another parameter along the four shortest members. This creates a new possible reference for the location of the interstitial parts other than one of the two ends of the members. If enough parameters are stipulated, subtler deviations in every cast can be achieved within the framework of a single alternating instance. The results show the dramatic effect upon the concrete form of maximising just one more parameter linked to all of the interstitial parts. An obvious extension of this idea was to ‘unlock’ one of the primary points dictating each instance to create a free parameter, also creating the potential for every increment of the cast structure to be disparate if desired. This

increases the complexity of the cast, but liberates the system from the monotony of identical odd and even instances.

Both of these systems increase the number of non-standard parts, but this explosion in complexity can be absorbed by producing templates from the digital model, ready to be cut by a machine. The limitless possibilities created by this alternative cannot necessarily be cast physically, but this does not inhibit exploring them digitally.

Physical Characteristics of the System

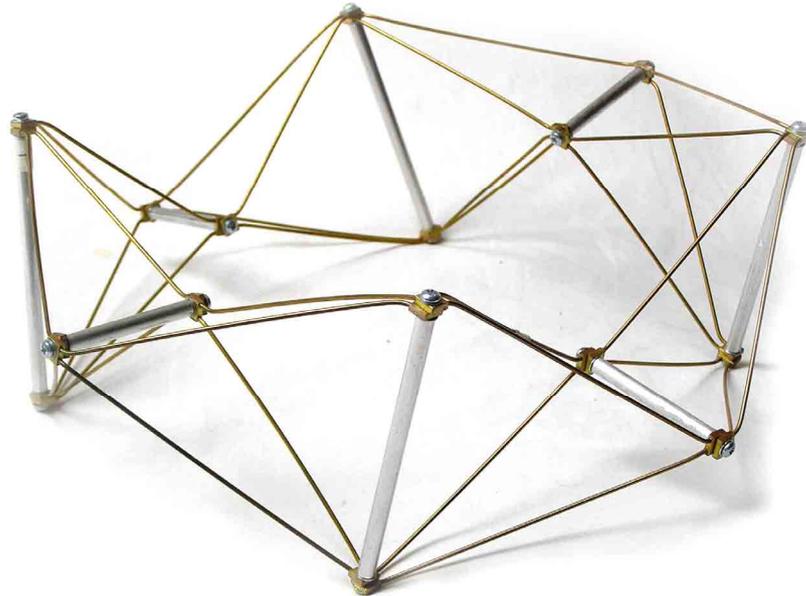


Figure 24 - The refined physical prototype.

Revision of the first physical prototype involved its remanufacture to integrate four new properties (Fig. 24):

- assume a fixed position without restricting movement;
- unwrap and permit easy release of the form from any cast increment;
- interlock with previous and subsequent instances;
- attach and remove standard and non-standard faces;
- cast a full scale column.

Some of these were foreseen before the digital model was built, but some, such as the design of the interstitial parts, could not have been confidently resolved without reference to it.

The redesign emphasises the complexities of the system, but the solutions exploit the advantages over casting with traditional methods. The minimal members and mechanisms (each frame is constructed from 112 manufactured components (Fig. 25)) would never cope with the lateral pressure exerted by a deep single pour, but the process of incremental casting reduces these forces. It was crucial however that the method of assembling multiple instances remained identical to the digitally modelled procedure (Figures 26-27).



Figure 25 - Workshop photos of machine assisted fabrication.

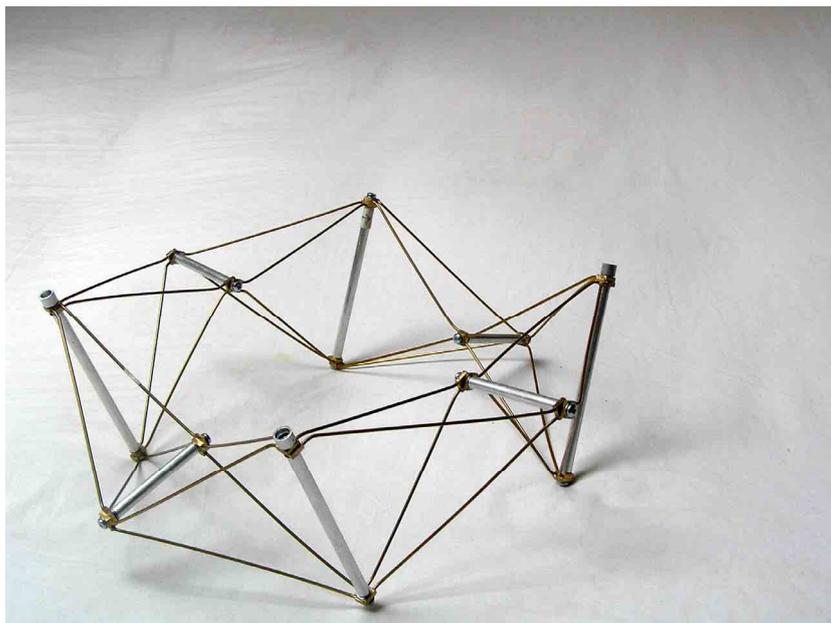


Figure 26 - The first instance of the second prototype with locating washers.

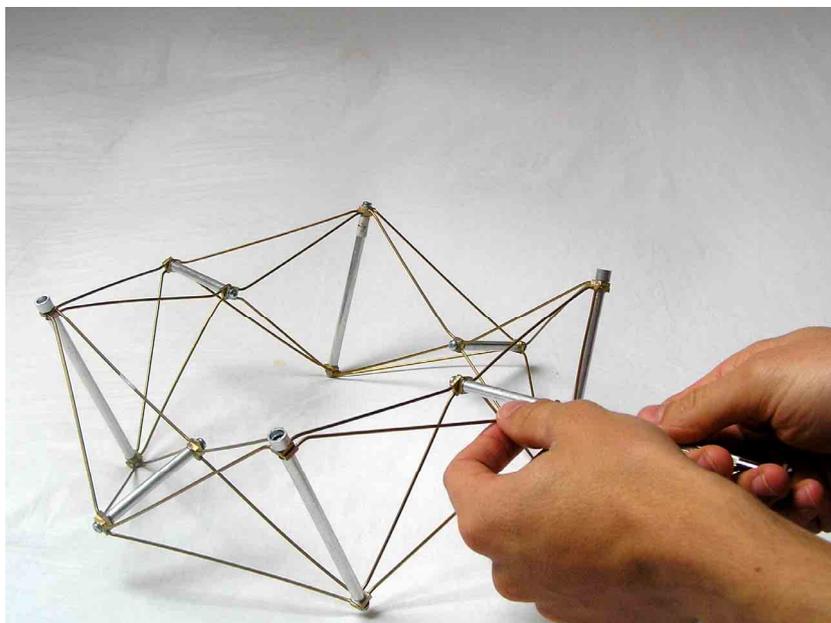


Figure 27 - Tightening the tension screws to fix the first instance.

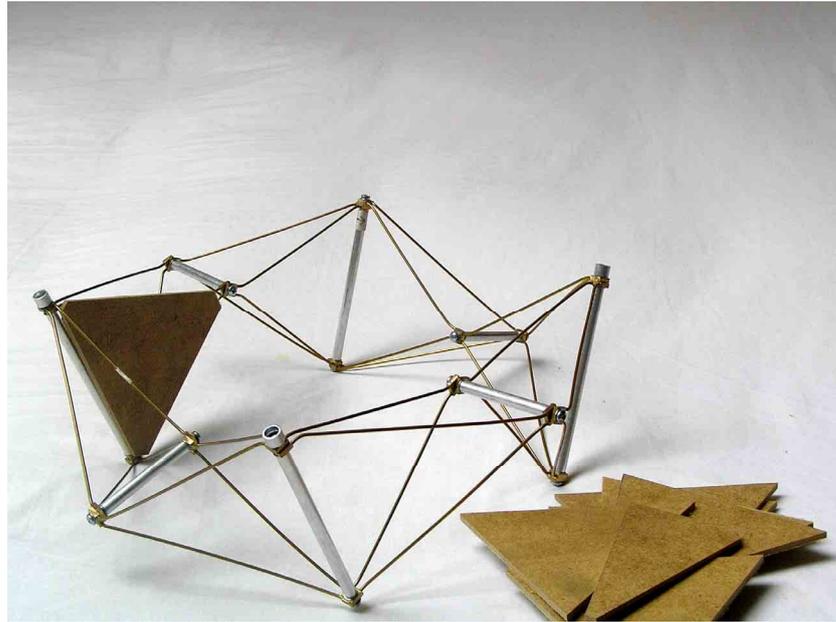


Figure 28 - Attaching faces to the first instance.

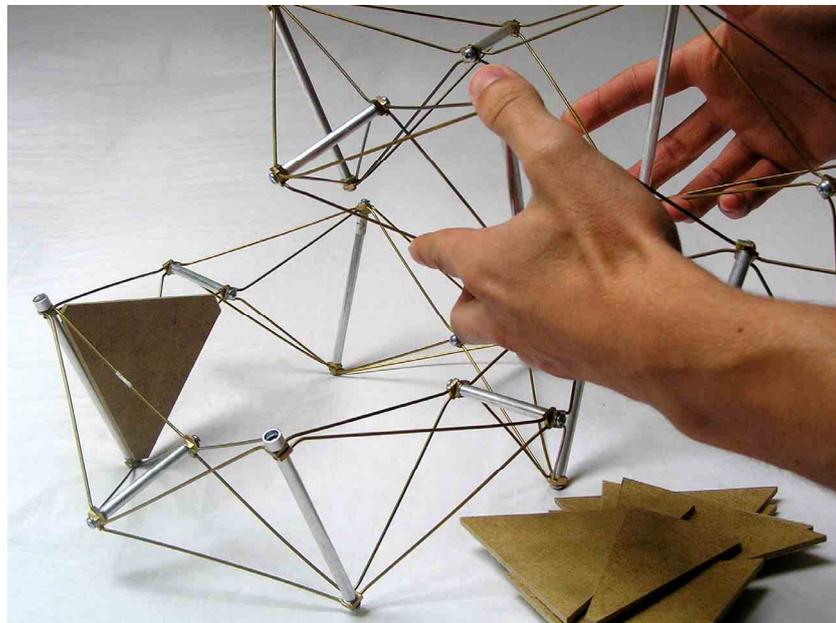


Figure 29 - Matching the primary points of the secondary instance with the secondary points of the first instance.

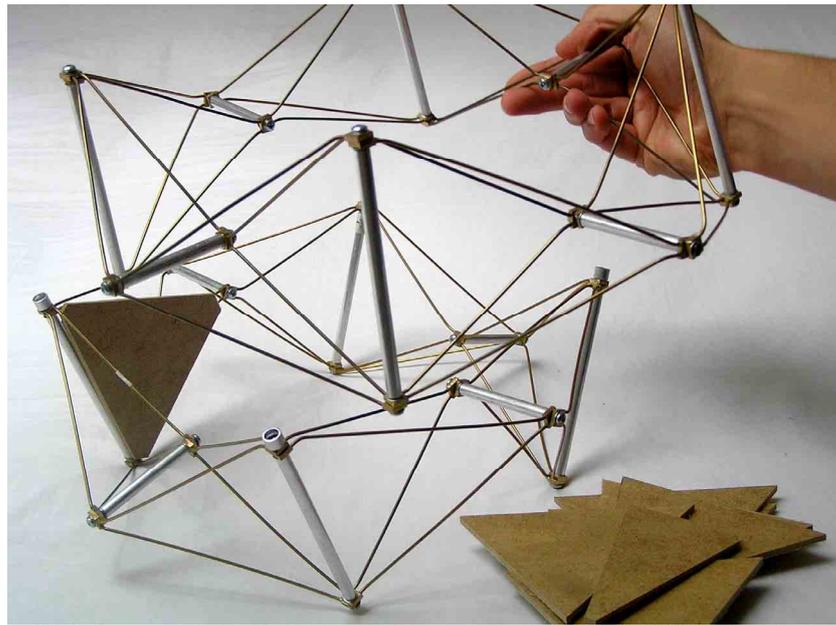


Figure 30 - Locating the second instance of the system.

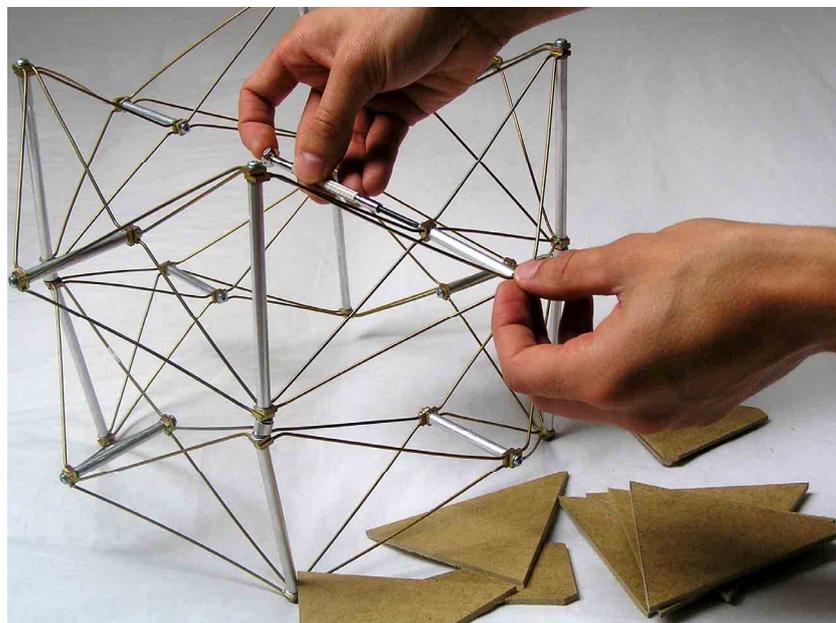


Figure 31 - Tightening the tension screws to fix the second instance.

Other Approaches

It has been reasoned that neither a constraint nor a pure graph-dependent approach was appropriate to model all of the properties of the system, so it cannot be suggested that either be trialled in isolation and compared.

It is envisaged that within an overall graph-based dependency model, the designer will be able to define (where appropriate) collections of components whose behavior will be controlled by constraint solvers (p.9 Aish 2005).

A need to control some components with constraint solvers within a graph-based dependency model has been documented, but these characteristically disparate approaches have not yet been hybridised. Even though it is speculated that bi-directional relationships might be established within a graph that is directed, whether these two are compatible remains uncertain.

MicroStation

It was possible to methodically construct some permutations in non-parametric architectural software if the values of all variables were fixed. To explore the possibilities of the system in MicroStation is unfeasibly slow, but for refining the details of mechanisms and other augmentations prior to fabrication, vector graphics are still valid. Watching how the values of different graph-variables instantly propagate through a system is useful to understand and design the details which permit its dynamism. Documenting them for fabrication however, is currently best accomplished by extracting the basic geometries from GenerativeComponents and embellishing them in MicroStation.

Scripting components to illustrate all of the joints and their mechanical operation is a convoluted process, although it would have been advantageous to update GenerativeComponents with details of the mechanisms and effectively re-prototype the system digitally. This 'digital prototyping' might simply involve creating extra construction geometry to model the paths of new and slightly relocated components.

Dimension Driven Design (DDD)

If a constraint solver such as DDD had been used, then just the distances between the connected points would be sufficient to calculate the infinite permutations of one instance of the system (with infinite time). It is possible to explore a large number of these instances hypothetically, but many of them may not have been useful in reality. It is this lack of control or predictability that is at the core of the problem of implementing a constraint manager or solver. Neither the accessibility of an approach, nor its propensity for simple descriptions and simple inputs can guarantee a simple output.

Examples of emergent behaviour which contravene this assumption are numerous in the natural world (ant colonies) and artificial worlds which mimic it (Conway's Game of Life). Knowing so many possible configurations is liberating, but this is disadvantageous for fabrication purposes. Apparently complex structures are frequently deceptively simple. This reduces building costs and permits human assembly. Employing a constraint solver can ironically result in an insufficiently constrained output. The distance between the nodes of the prototype is an inadequate description to calculate the movement of the prototype through space when the application of the model is specific and the many limitations of building it in the real world must be acknowledged and applied.

The most important constraints for this project are not the sizes of the members, but those which pertain to the implementation of the prototype functioning as the formwork for casting concrete. In addition to minimising costs and complexity of assembly, these include, but are not limited to, the horizontal levelling of the concrete as it is influenced by gravity, the ability of any usable configuration to contain the curing mixture and ultimately the strict hierarchy that is imposed by an incremental process of vertical casting. It is these attributes of the system which remain the most important constraints, not the dimensions of its components.

Limitations

Casting is a procedure constrained by the material levelling horizontally. This property of the process justified the digital modelling of a system which creates each new instance of its components on a plane parallel to the one upon which the last instance was created. This precludes the possibility that the central axis of an incrementally cast structure might be curved to produce an arch (a structure which is neither horizontal nor vertical and neither column nor beam) and that this resulting curved column might be cast cohesively to form a thin-shell roof or dome. The horizontal levelling could be overcome by designing an appropriate mix and pumping it into a more enclosed mould. If the digital model and physical mechanism could be derestricted in these ways then an adaptable formwork, with the ability to cast an entire shelter, would be a self-sufficient system of construction.

Although the system was digitally modelled to accommodate the three degrees of freedom of the physical prototype, only two of them were exploited to reduce the number of parameters. The third degree of freedom is possible only when the members between the primary and secondary points are tangential to the horizontal plane. In this specific scenario the system can produce infinite rhombi, but the one primary point which controls these possibilities must always be located on the circumference of a circle constructed from the nearest other primary point. This is very difficult to achieve with the simple inputs (X, Y & F) that have been explored in this thesis. This property of the prototype remains interesting, but these permutations are not as simple to describe as those which exhibit just two degrees of freedom.

Further Work

It is obvious that the revised physical system should be tested against its digital ideal to expose any discrepancies. It is planned that the system will be transported to the first F2F continuum workshop in Lyon to cast a concrete column using Vicat rapid setting concrete. The redesign of the system must be assimilated into the digital model. As its construction was machine assisted it was possible to produce parts of the model with tolerances that were less than half a millimetre, but the mechanisms still do not quite enable the members to meet at the same point. These disparities are inevitable, but consistent enough to be modelled digitally.

These small errors accumulate through such a large number of parts, further reducing the accuracy of the correlation between the two. Assembly was completely unassisted by machines. These disparities are inconsistent and harder to measure so they are more difficult to build into the digital model. It is simple to eradicate human error and work to more stringent tolerances if the digital model had been fabricated entirely by a machine. Any significant discrepancies (beyond expected tolerances) which endure both of these refinements would be worthy of closer inspection and detailed analysis.

In summary, the system produced by this study is comparable to Rubin's vase: an optical illusion that is interpretable as the silhouette of a vessel that shares the middle axis of the image, but equally the two faces which flank it. Although the ability of the system to adapt to casting concrete is the focus of this research, it is equally possible to see the frame of the system itself as the focus of further study. It might cast space, not concrete, and create shelter in ways familiar and unfamiliar to contemporary construction.

Conclusion

Sequencing and mechanising the fabrications of buildable structures demands a hybridisation of approaches to digital modelling. Some more complex constructions cannot be modelled purely with hierarchies or constraints. When the sequencing of a mechanism is crucial to the process of construction, it may be possible to dissect the real process of assembly and employ additional construction geometries to solve constraints with a graph-based dependency model.

The focus of this thesis has been upon modelling a specific process of construction. A new parametric formwork has been prototyped to incrementally cast concrete structures as an alternative to single pour methods which waste money and material by using more material for the facing (to describe the entire form at once) and frame (to brace the faces for the lateral pressure of a large volume of concrete). The proposed system produces triangulated structures which are deceptively repetitive and improve the economy of the process by enabling the reuse of small faces again and again. These can be supported by a minimal frame which would break if subjected to the lateral pressure exerted at the base of the concrete structures it is capable of phasing, but must only be strong enough to contain much shallower pours. These properties of the system create the illusion that it is emergent.

The design of this system involved adjusting multiple instances of the formwork so that the primary points of each one are set according to the previous instance. The behaviour of a single prototyped instance of the formwork had to first be correlated with its digital equivalent. Its parametric nature permits the reduction of permutations with two degrees of freedom to

just two variables describing three primary points which not only dictate the location of the first instance of the adaptable formwork in space, but also every subsequent instance. Additional parameters can be added to control the offset of the faces which span between instances to vary the number of facets cast in the concrete. A less rigid adhesion to the process of constructing the formwork was explored by freeing one of the points controlling each instance. This simultaneous alteration of the trajectory of the column suggests the invention of a complete building system if the central axis could be, not just linearly modified, but curved too. It has been suggested that the resulting explosion in non-standard parts could be countered by cutting them directly from the digital model.

The problem arising from this necessity however is the minimisation of discrepancies between the digital ideal and physical reality of the system so that physical possibilities might be digitally explored. What has been shown is that certain physical processes of construction with strict procedures, but bi-directional aspects can be parametrically modelled digitally in GenerativeComponents without a constraint solver controlling components. Simple dependency relationships may suffice.

Bibliography

Aish, R., 2005, 'Introduction to Generative Components',
<http://www.bentley.com/en-US/promo/flash/CIG+Resources.html> (accessed 5th June 2008).

Burry, M., 1993, *Expiatory Church of the Sagrada Familia*, (Singapore; London: Phaidon Press, 1993).

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

Fossa, K.T., 2001, 'Slipforming of Vertical Concrete Structures',
http://www.diva-portal.org/diva/getDocument?urn_nbn_no_ntnu_diva-42-1__fulltext.pdf (accessed 10th September 2008).

Frazer, J., 1995, *An Evolutionary Architecture*,
<http://www.aaschool.ac.uk/publications/ea/intro.html> (accessed 10th September 2008).

Killian, A., 'Fabrication of partially double-curved surfaces out of flat sheet metal through a 3d puzzle approach',
<http://designexplorer.net/projectpages/prototyping.html> (accessed 5th June 2008), presented at *Arcadia 2003*.

Legenza, K., 2007, "Twisting Steel and Bending the Imagination", *Structure Magazine*, November 2007, pp.47-49.

McAdam, P.S. & Lee, G.W., 1997, *Formwork*, (London: E & FN Spon, 1997).

Pawley, M., 1990, *Design Heroes: Buckminster Fuller*, (London: Grafton, 1992).

Peurifoy, R. L. & Oberlender, G. D., 1996, *Formwork for Concrete Structures*, (McGraw-Hill, 1996, 3rd ed.).

Prosalidou, E. & Hanna, S., 2007, 'A Parametric Representation of Ruled Surfaces', <http://www.sean.hanna.net/publications/parametricRuledSurf.pdf> (accessed 10th September 2008).

Richardson, J.G., 1977, *Formwork Construction and Practice*, (London: E & FN Spon, 1978).

Sass, L., 2004, 'Design for Self Assembly of Building Components using Rapid Prototyping', <http://ddf.mit.edu/papers/index.html> (accessed 10th September 2008).

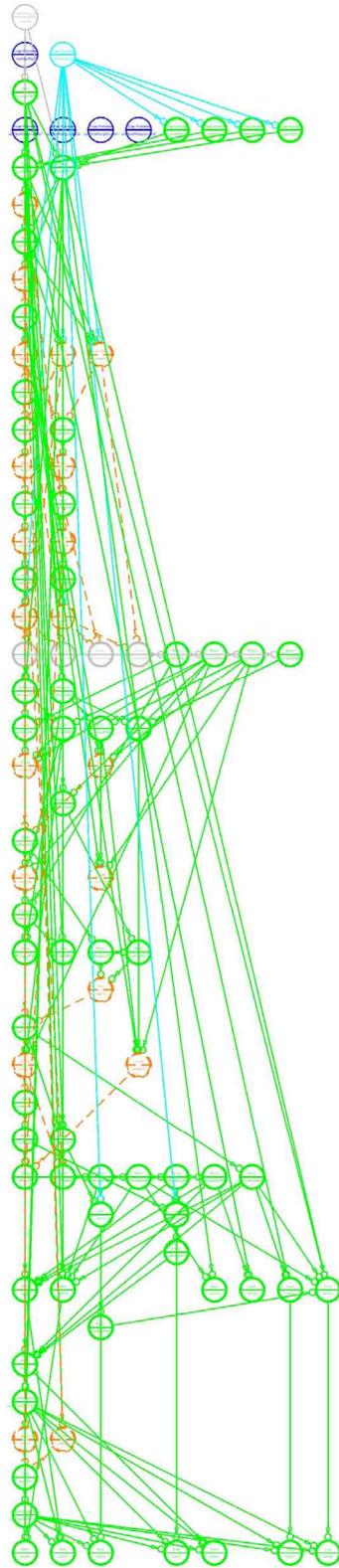
Sass, L., 2005, 'Wood Frame Grammar: CAD scripting a wood frame house', <http://ddf.mit.edu/papers/index.html> (accessed 10th September 2008).

Sass, L. & Botha, M., 2006, 'The Instant House: A production system for construction with digital fabrication', <http://ddf.mit.edu/papers/index.html> (accessed 10th September 2008).

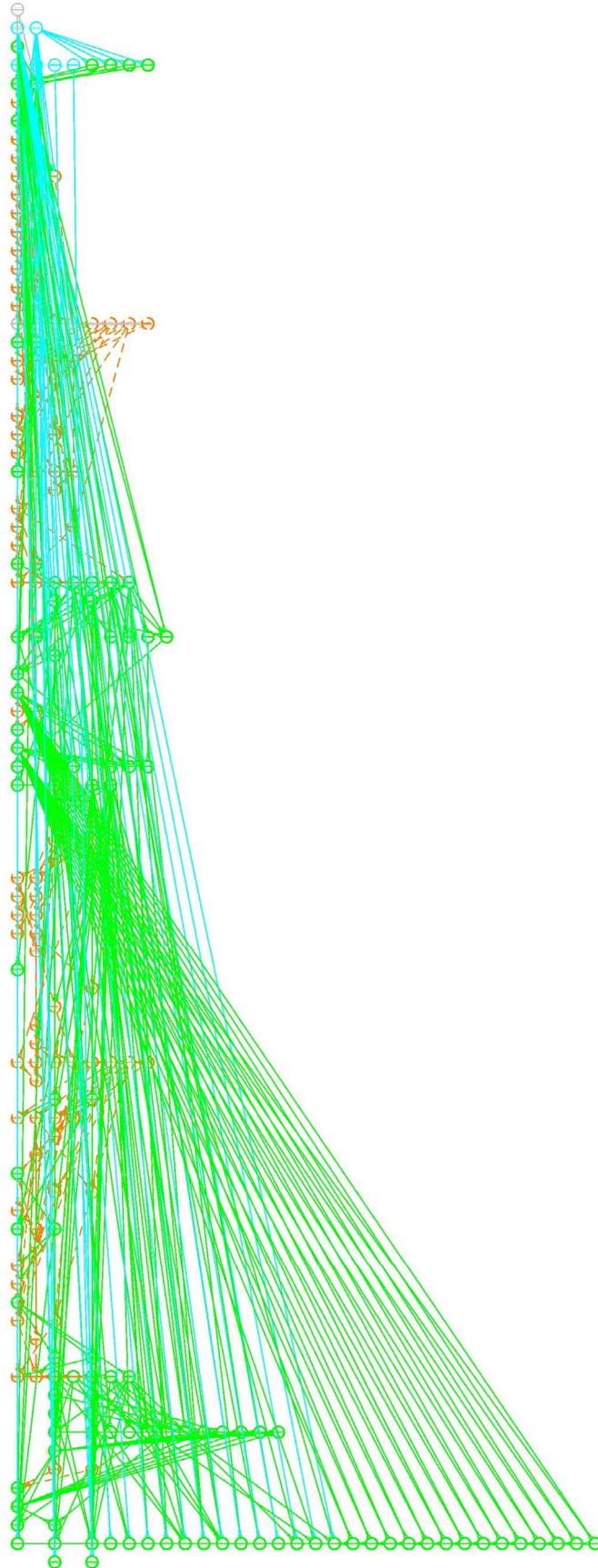
Wynn, A. E. & Manning, G.P., 1926, *Design and Construction of Formwork for Concrete Structures*, (George Berridge & Co. Ltd., 1973, 5th ed.).

Appendix I

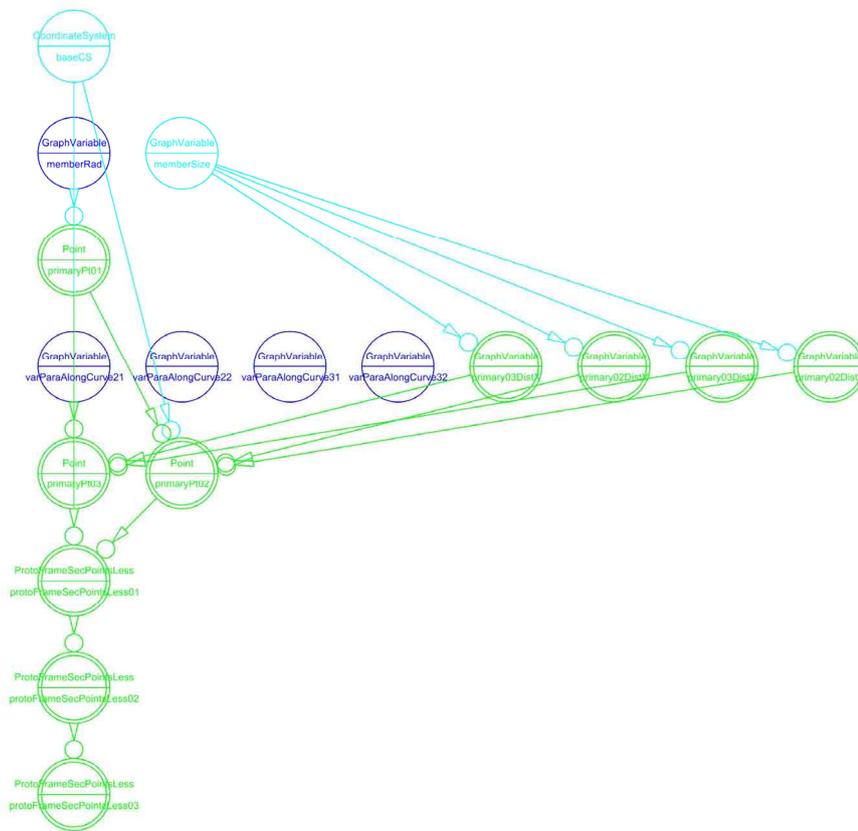
Symbolic Graph [a]



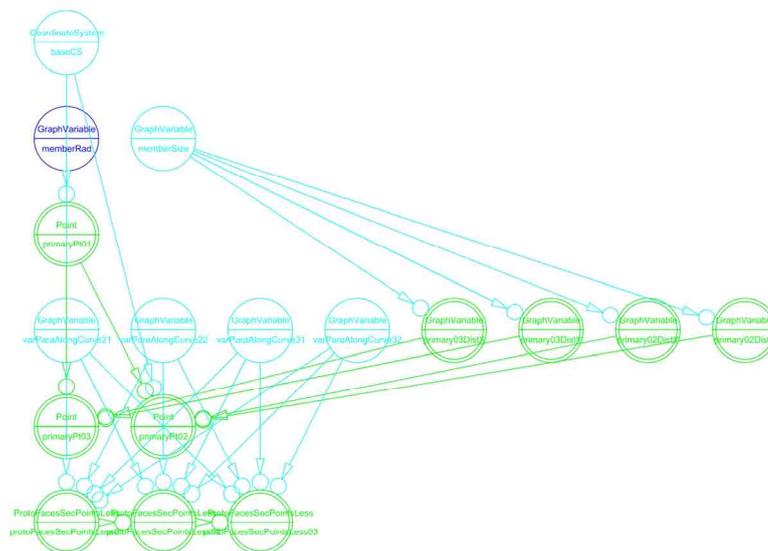
Symbolic Graph [b]



Symbolic Graph [c]



Symbolic Graph [d]



Appendix II

Code segment [a]

transaction script "conditional script to choose from possible meeting points"

```
{
  if (Distance(possMeetPt01[0],primaryPt03) >=
Distance(possMeetPt01[1],primaryPt03))
  {
    Point corrMeetPt01 = new Point ("corrMeetPt01");
    corrMeetPt01.CentroidOfSet ({possMeetPt01[0],possMeetPt01[0]});
  }
  else if (Distance(possMeetPt01[0],primaryPt03) <
Distance(possMeetPt01[1],primaryPt03))
  {
    Point corrMeetPt01 = new Point ("corrMeetPt01");
    corrMeetPt01.CentroidOfSet ({possMeetPt01[1],possMeetPt01[1]});
  }
  if (Distance(possMeetPt02[0],primaryPt02) >=
Distance(possMeetPt02[1],primaryPt02))
  {
    Point corrMeetPt02 = new Point ("corrMeetPt02");
    corrMeetPt02.CentroidOfSet ({possMeetPt02[0],possMeetPt02[0]});
  }
  else if (Distance(possMeetPt02[0],primaryPt02) <
Distance(possMeetPt02[1],primaryPt02))
  {
    Point corrMeetPt02 = new Point ("corrMeetPt02");
    corrMeetPt02.CentroidOfSet ({possMeetPt02[1],possMeetPt02[1]});
  }
}
```

Code segment [b]

transaction script "conditional script to choose real secondary point"

```
{
  if (Distance(primaryPt03, refPrimaryPt03B) == Distance(refPrimaryPt03A,
refPrimaryPt03B))
  {
    Point corrRealSecPt01 = new Point ("corrRealSecPt01");
    corrRealSecPt01.CentroidOfSet ({possRealSecPt01,possRealSecPt01});
  }
  else if (Distance(primaryPt03, refPrimaryPt03B) <
Distance(refPrimaryPt03A, refPrimaryPt03B))
  {
    Point corrRealSecPt01 = new Point ("corrRealSecPt01");
    corrRealSecPt01.CentroidOfSet ({possRealSecPt01[1],possRealSecPt01[1]});
  }
  else if (Distance(primaryPt03, refPrimaryPt03B) >
Distance(refPrimaryPt03A, refPrimaryPt03B))
  {
    Point corrRealSecPt01 = new Point ("corrRealSecPt01");
    corrRealSecPt01.CentroidOfSet ({possRealSecPt01[2],possRealSecPt01[2]});
  }
}
```

Code segment [c]

```

transaction script "conditional script to choose correct faces and intersecting points"
{
    if (Distance(primaryPt01, primaryPt03) < 2*(Sqrt((memberSize*memberSize) -
(memberSize/2*memberSize/2))))
    {
        Plane corrFace01 = new Plane ("corrFace01");
        corrFace01.ByOriginXYPoints(primaryPt03, primaryPt01, realSecPt01[0]);
        Plane corrFace02 = new Plane ("corrFace02");
        corrFace02.ByOriginXYPoints(primaryPt02, primaryPt01, realSecPt01[0]);
        Point corrFacePt01 = new Point ("corrFacePt01");
        corrFacePt01.AtPlaneCurveIntersection(corrFace01, {circle05,circle11});
        Point corrFacePt02 = new Point ("corrFacePt02");
        corrFacePt02.AtPlaneCurveIntersection(corrFace02, {circle04, circle10});
    }
    else if (Distance(primaryPt01, primaryPt03) > 2*(Sqrt((memberSize*memberSize) -
(memberSize/2*memberSize/2))))
    {
        Plane corrFace01 = new Plane ("corrFace01");
        corrFace01.ByOriginXYPoints(primaryPt03, primaryPt01, realSecPt01[1]);
        Plane corrFace02 = new Plane ("corrFace02");
        corrFace02.ByOriginXYPoints(primaryPt02, primaryPt01, realSecPt01[1]);
        Point corrFacePt01 = new Point ("corrFacePt01");
        corrFacePt01.AtPlaneCurveIntersection(corrFace01, {circle05,circle11});
        Point corrFacePt02 = new Point ("corrFacePt02");
        corrFacePt02.AtPlaneCurveIntersection(corrFace02, {circle04, circle10});
    }
}

```

Code segment [d]

```

transaction script "conditional script to choose real secondary points"
{
    if (Distance(corrFacePt02[0][1],vertRefPoint) < Distance(corrFacePt02[0][0],vertRefPoint))
    {
        Point corrRealMeetPt01 = new Point ("corrRealMeetPt01");
        corrRealMeetPt01.CentroidOfSet({corrFacePt02[0][1],corrFacePt02[0][1]});
    }
    else if (Distance(corrFacePt02[0][1],vertRefPoint) > Distance(corrFacePt02[0][0],vertRefPoint))
    {
        Point corrRealMeetPt01 = new Point ("corrRealMeetPt01");
        corrRealMeetPt01.CentroidOfSet({corrFacePt02[0][0],corrFacePt02[0][0]});
    }
    if (Distance(corrFacePt02[1][0],vertRefPoint) < Distance(corrFacePt02[1][1],vertRefPoint))
    {
        Point corrRealSecPt02 = new Point ("corrRealSecPt02");
        corrRealSecPt02.CentroidOfSet({corrFacePt02[1][0],corrFacePt02[1][0]});
    }
    else if (Distance(corrFacePt02[1][0],vertRefPoint) > Distance(corrFacePt02[1][1],vertRefPoint))
    {
        Point corrRealSecPt02 = new Point ("corrRealSecPt02");
        corrRealSecPt02.CentroidOfSet({corrFacePt02[1][1],corrFacePt02[1][1]});
    }
    if (Distance(primaryPt01, primaryPt03) < Sqrt((memberSize*memberSize) - (memberSize/2*memberSize/2)))
    {
        if (Distance(corrFacePt01[0][1],vertRefPoint) < Distance(corrFacePt01[1][1],vertRefPoint))
        {
            Point corrRealSecPt02 = new Point ("corrRealSecPt02");
            corrRealSecPt02.CentroidOfSet({corrFacePt01[0][1],corrFacePt01[0][1]});
        }
        else if (Distance(corrFacePt01[0][1],vertRefPoint) > Distance(corrFacePt01[1][1],vertRefPoint))
        {
            Point corrRealSecPt02 = new Point ("corrRealSecPt02");
            corrRealSecPt02.CentroidOfSet({corrFacePt01[1][1],corrFacePt01[1][1]});
        }
        if (Distance(corrFacePt01[1][0],vertRefPoint) < Distance(corrFacePt01[1][1],vertRefPoint))
        {
            Point corrRealMeetPt02 = new Point ("corrRealMeetPt02");
            corrRealMeetPt02.CentroidOfSet({corrFacePt01[1][0],corrFacePt01[1][0]});
        }
        else if (Distance(corrFacePt01[1][0],vertRefPoint) > Distance(corrFacePt01[1][1],vertRefPoint))
        {
            Point corrRealMeetPt02 = new Point ("corrRealMeetPt02");
            corrRealMeetPt02.CentroidOfSet({corrFacePt01[1][1],corrFacePt01[1][1]});
        }
        Point newCorrRealSecPt01 = new Point ("newCorrRealSecPt01");
        newCorrRealSecPt01.CentroidOfSet({corrRealSecPt01, corrRealSecPt01});
    }
    else if (Distance(primaryPt01, primaryPt03) > Sqrt((memberSize*memberSize) - (memberSize/2*memberSize/2)))
    {
        if (Distance(corrFacePt01[0][0],vertRefPoint) < Distance(corrFacePt01[0][1],vertRefPoint))
        {
            Point corrRealMeetPt02 = new Point ("corrRealMeetPt02");
            corrRealMeetPt02.CentroidOfSet({corrFacePt01[0][0],corrFacePt01[0][0]});
        }
    }
}

```

```

else if (Distance(corrFacePt01[0][0],vertRefPoint) > Distance(corrFacePt01[0][1],vertRefPoint))
{
    Point corrRealMeetPt02 = new Point ("corrRealMeetPt02");
    corrRealMeetPt02.CentroidOfSet({corrFacePt01[0][1],corrFacePt01[0][0]});
}
if (Distance(corrFacePt01[1][0],vertRefPoint) < Distance(corrFacePt01[1][1],vertRefPoint))
{
    Point corrRealSecPt03 = new Point ("corrRealSecPt03");
    corrRealSecPt03.CentroidOfSet({corrFacePt01[1][0],corrFacePt01[1][0]});
}
else if (Distance(corrFacePt01[1][0],vertRefPoint) > Distance(corrFacePt01[1][1],vertRefPoint))
{
    Point corrRealSecPt03 = new Point ("corrRealSecPt03");
    corrRealSecPt03.CentroidOfSet({corrFacePt01[1][1],corrFacePt01[1][1]});
}
if (Distance(realSecPt01[0],vertRefPoint) < Distance(realSecPt01[1],vertRefPoint))
{
    Point newCorrRealSecPt01 = new Point ("newCorrRealSecPt01");
    newCorrRealSecPt01.CentroidOfSet({realSecPt01[0],realSecPt01[0]});
}
else if (Distance(realSecPt01[0],vertRefPoint) > Distance(realSecPt01[1],vertRefPoint))
{
    Point newCorrRealSecPt01 = new Point ("newCorrRealSecPt01");
    newCorrRealSecPt01.CentroidOfSet({realSecPt01[1],realSecPt01[1]});
}
}
}
}

```

Code segment [e]

```

transaction modelBased "draw frame"
{
    feature line05 GC.Line
    {
        StartPoint          = corrRealMeetPt02;
        EndPoint              =
    {corrRealSecPt03,primaryPt03,primaryPt01,newCorrRealSecPt01};
    }
    feature line06 GC.Line
    {
        StartPoint          = corrRealMeetPt01;
        EndPoint              =
    {corrRealSecPt02,primaryPt02,primaryPt01,newCorrRealSecPt01};
    }
    feature line20 GC.Line
    {
        StartPoint          = subMeetPt01;
        EndPoint              =
    {corrRealSecPt02,primaryPt02,primaryPt01,newCorrRealSecPt01};
    }
    feature line21 GC.Line
    {
        StartPoint          = subMeetPt02;
        EndPoint              =
    {corrRealSecPt03,primaryPt03,primaryPt01,newCorrRealSecPt01};
    }
    feature line22 GC.Line
    {
        StartPoint          = primaryPt01;
        EndPoint              = newCorrRealSecPt01;
    }
    feature line23 GC.Line
    {
        StartPoint          = primaryPt02;
        EndPoint              = corrRealSecPt02;
    }
    feature line24 GC.Line
    {
        StartPoint          = primaryPt03;
        EndPoint              = corrRealSecPt03;
        LineWeight           = 2;
    }
}

```