Robert Aish *is Director of Research at Bentley Systems and is responsible for the development of a new parametric design software called 'GenerativeComponents'. He is also a founder member of the SmartGeometry Group who are concerned with the wider educational implications of the teaching and practice of parametric design. In this context, Robert Aish is currently working with a number of leading architectural practices including Foster and Partners, Morphosis, KPF, Grimshaw, NBBJ and ONL, and a number of Schools of Architecture, including the Architectural Association, MIT, Georgia Tech. and the Technical University of Delft.*

From Intuition to Precision

Design has been described as making inspire decisions with incomplete information. True, we may use prior knowledge, we may even think we understand the causalites 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.

So how can we progress from intuition to precision? What abstractions can we use to represent, externalize and test the concepts involved? How can we augment the cognitive processes? How can we record the progression of ideas? And, how do we know when we have arrived?

Design has a symbiotic relationship with geometry. There are many design issues that are independent of any specific configurations. We might call these "pre-geometric" issues. And having arrived at a particular configuration, there may be many material interpretations of the same geometry. We might call these "post-geometric" issues. But geometry is central to design, and without appropriate geometric understanding, the resulting design will be limited. Geometry has two distinct components, one is a formal descriptive system and the other is a process of subjective evaluation. This duality is paradoxical and leads to the following interaction:

"This is the shape I have sketched, but how to I formally describe it" versus "This is the formal shape description system I am using, how do I harness and control this system to create the shape I like". We would most probably agree that we would not want the formality to constrain exploration, while recognizing that without an understanding of the formal system such exploration will be difficult to realize as a physically constructible artifact.

We can characterise design as being different to craft (because the designer does not directly act on the material, but has an indirect, and arguably more powerful, way of controlling materialization). Similarly, we might characterise computational design as beng different to conventional design (because the designer is not directly drawing the geometry, but has an indirect, and arguably more powerful, way of controlling that geometry using computational design tools).

In this way we should view computational design as part of a normal progression in which the designer and the artifact are separated by an increasing number of levels of indirection, that in turn introduce higher levels of expression and control. Opponents of this may question whether introducing these levels of indirection is in fact progress, arguing that intuition and spontaneity will be inhibited with the increased remoteness between the designer and artifact. Happily these layers of indirection are not arranged linearly, but can be configured to form a closed loop. The advent of digitally controlled fabrication means that the 'geometrically aware' and 'computationally enabled' designer is as close to the materialization as in the original craft process, but with precision and control and the ability to explore variation which was previously unimaginable.

The question now is: what are the characteristics of computational design tools that facilitate this approach to design and what are the corresponding abstractions which need to be internalized and operationalised by designers? The essentially themes are:

- Geometry
- Composition
- Algorithmic thought

<u>Geometry</u>: We need to start with a fundamental understanding of geometric primitives: points, planes, coordinate systems, line arc, curves, surfaces and solids. We need to understand what the 'order' of a curve means, how curves and surfaces are parameterized. We need to understand geometric operations on these primitives: projection, intersection, union, difference, transformation. We need to use these primitives and operations to define relationships. We need to understand the stability of these geometric relationships under certain modifications and configurations.

With this understanding, we will have the opportunity to build 'long chain' dependencies which will create interesting geometric configurations. What is important is not the static configuration, but the way in which some change, for example to the location of a key point or parameter, can create alternative configurations.

What becomes apparent is that we are not designing the geometry of the artifact, but rather we are constructing a 'control rig', some geometry that will never be built or seen, but which indirectly controls what will be constructed and experienced. It is the development of this sense of 'indirection' or 'control through geometric dependency' which is emerging as a key design skill. By building and exercising these systems of geometric dependency we are able to explore variation in design, indeed to explore the solution space, and to discover and validate the configuration that will finally be constructed.

<u>Composition</u>: It is pretty rare to find a building which is a 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 to 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 'composi-

tion' and which allow these strategies to be developed and tested.

Algorithmic thought: At one level these is a desire to explore geometric subtleties which go beyond what 'hand-eye coordination' can deliver. At another level there is a desire to apply ideas of 'consistency' or controlled 'unpredictability' over large data sets, for example representing a building façade. Essentially this geometry cannot be drawn. It has to be computed. If it is to be computed, then there has to be an algorithm. To be original and to be in control, the designer has to understand, if not originate, his own algorithm, and know how to 'drive it' (i.e. know what are valid inputs and know how to interpret, verify and validate the results and know the limits to the solution space).

Does this mean that the designer of the future has to be a programmer? No, but it might help. Certainly, developing an ability to think algorithmically will emerge as a key design skill. But how can we encourage the development of these skills without demanding that designers become programmers? A potentially fruitful approach is to introduce the necessary logical formalism in very small doses. This has the important advantages for the designer in that he can discover the value of embedding 'logic' in his design model in a 'declarative' form without having to completely master all the constructs normally associated with a procedural programming language. But what is being learnt is not expressed in some 'cut-down' over simplified syntax, but uses established programming conventions, so that as the designer becomes more computationally expressive, he can build on these initial steps.

Most importantly, algorithmic design does not imply that subjectivity is out of the loop, or even that 'hand-eye coordination' is redundant. What we have facilitated is the ability for the designer to embed his design logic within an interactive design system which is driven by the designer's hand and evaluated by the designer's eye. This follows the fundamental precept of design, that of the combining intuition and precision into a single process and with the results of that process integrated and embodied in the same artefact.

At the foundation of computational design is the relationship between tools and skills. We have to match our tools to the concepts around which designers want to build their skills. Our expectation is that geometric skills, compositional skills and algorithmic skills will be the key to future design.

Fig. 1 shows a 3D model of a surface constructed using cross-sectional control curves, which in turn have been created on planes defined on a primary 'path' curve. Overlaid on the surface are a 2D array of points uniformly spaced in the parameter space of the surface, and then overlaid on the points are a series of quadrilaterals. This configuration might represent the design of an 'idealised' roof surfaces which must be panelized into fabricatible sub-components. In this model the following variations are possible: (A) The poles of the path curve can be moved in Cartesian space (B) the position of the planes on the path curve can be move in 1D parametric space (along the path curve) (C) the poles of the cross-sectional curves can be moved in the 2D planar space (D) the number and spacing of the points on the surface can be defined in the surface 2D parameter space (E) various alternative 'lacing' option are available to use the points on the surface to populate either planar or non-planar quadrilaterals panels or triangular panels (F) the order of the path curve and cross sectional curves can be varied. This is a complex system of relationships. It is unlikely that the designer can define and manage these relationships without some computational support to externalise and record his design intent in an editable and re-executable form.

Fig. 2 show the Symbolic model, which externalizes and presents these relationships in an explicit graphical firm. This is a system of 'multiple representation', in which for each 'component' of the model there is a direct correspondence between the graphic and symbolic representations. The designer can

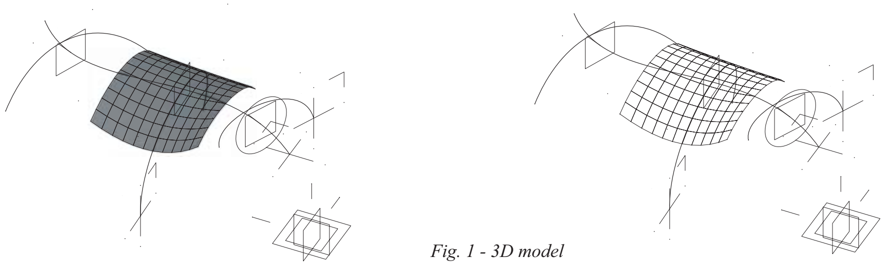


Fig. 1 - 3D model

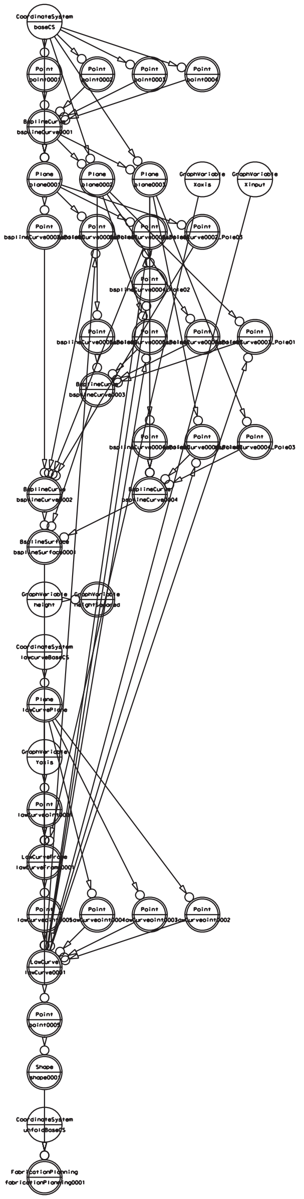


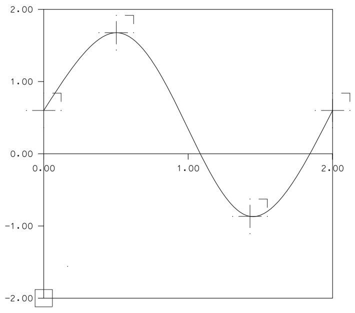
Fig. 2 - Symbolic model

observe and modify these relationships by locating a ‘component’ of the model either in its 3D graphical form (in the 3D model) or in its symbolic representation (in the Symbolic model). Note, that a ‘component’ may be an explicit geometric element with a graphical representation, such as a point, plane, curve, surface, solid. Alternatively, a component may be more abstract, such as a variable, expression, conditional statement, or even a user defined script. It may not have a graphical representation, but nevertheless it still contribute to the ‘logical’ system of relationships which will result (finally) in some geometric expression.

In Fig 1. we showed a 3D model with explicit control points, used to define and manipulate the dependent curves and planes, using familiar techniques of ‘direct manipulation’. However we are not restricted to this approach. Sometimes a designer wishes to maintain relationships which are essentially geometric in nature, but the controlling geometry is not necessarily of the same dimensionality as other aspects of the design model, and may not be defined in the same ‘model space’ as the resulting design. In this example, the main surface model is defined in 3D geometry. However, let us suppose that the designer wants to defined the profile (or ridge line) of this surface in 2D, independent of the main 3D model. This would be as if (A) the path curve was unfolded to be a line (B) the ridge line was defined as an offset to this line (C) the path curve was returned to its original configuration (D) the ridge line was similarly transformed. We can achieve this result using a ‘Law Curve’ model shown in Fig. 3. A Law curve is essentially a geometrically defined ‘function’, which returns values for Y (the dependent variable) given a range of values for X (the independent variable) and a curve that defines the relationship between X and Y. In this case, the values of X represent the distance along the path curve and the values of Y represent the height of the poles of the cross section curves controlling the surface.

Design is not only about specifying and evaluating the resulting configuration, but also anticipating how that configuration can be materialized. The non-planar quadrilateral panels are a case in point. Fig 4. show how the array of panels can be unfolded into a series of planar strips, with each strip having an external cutting profile and each panel defined by internal ‘scribing’ lines. The choice of lines colour for the cutting and scribing lines can be matched to the power setting of a standard laser cutter.

The significance of this example is that it shows how the designer has achieved two important advances: (1) he has designed his own design tool (and indeed created his own GUI) and (2) he has formalized and externalized his design process, in a form which is understandable, editable and re-executable. So having defined this process, the designer



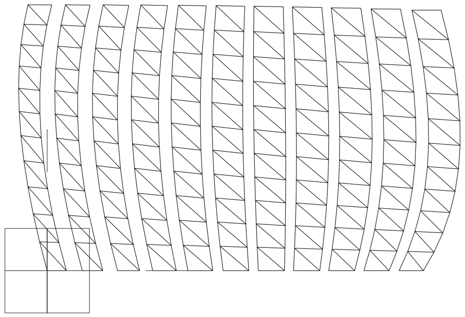


Fig. 3 - Law curve

Fig. 4 - Unfolded planar strips

can now explore variations within the solution space, not in some rigid parametric way, but by using an intuitive process of 'direct manipulation' and 'hand-eye coordination'. Here, the designer can graphically select and manipulate one of the control points of the Law curve model and observe: (A) the law curve update (B) the cross section curves update (C) the surface update (D) the points on the surface update (E) the quadrilateral panels on the surface update and (F) the planar unfolded fabrication model update. The whole process is being intuitively controlled in dynamics and at the same time the designer is closer to the materialization of his design than at any stage since the craft era. But to arrive at this combination of intuition and control, the designer had to be skilled in the logic of design, in order to define and refine the complex system of geometric, algebraic and logical relationships which is the essential foundation of this process. Ultimately what is of interest is not just the results that can be achieved, but that 'parametric design', as exemplified by GenerativeComponents, encourages designers to find expression through the exercise of a unique range and combination of ideas and skills.