Teaching Parametric Design in Code and Construction.

Sean Hanna
Alasdair Turner
Bartlett School of Graduate Studies,
University College London
email: s.hanna@cs.ucl.ac.uk

Parametric and generative methods of design, often incorporating explicit computer programming, represent a working method based more on procedure than geometric form. This paper reviews a recent workshop held at the Bartlett School of Architecture, using Bentley's Generative Components software to teach parametric modelling, with a focus on digital fabrication of the projects. We propose that in the design studio of the future, these procedural techniques may help to forge a closer connection between the process of form generation and the real, constructed design in one of two ways: either in how it is built or in how it performs.

Introduction

Automated manufacturing processes with the ability to translate digital models into physical form promise both an increase in the complexity of what can be built, and through rapid prototyping, a possibility to experiment easily with tangible examples of the evolving design. In addition, the increasing literacy of designers in computer languages offers a new range of techniques through which the models themselves might be generated. (Loukissas and Sass 2004) This paper reviews the results of an integrated parametric modelling and digital manufacturing workshop at the Bartlett School of architecture, combining participants with a background in computer programming with those with a background in fabrication. Its aim was both to encourage collaboration in a domain that overlaps both backgrounds, as well as to explore the ways in which the two working methods naturally extend the boundaries of traditional parametric design. The types of projects chosen by the students, the working methods adopted and progress made will be discussed in light of future educational possibilities, and of the future direction of parametric tools themselves.

Bentley's Generative Components (GC) software was used as the primary vehicle for the workshop design projects. Built within the Microstation framework, it enables the construction of a parametric model at a range of different interfaces, from purely graphic through to entirely code based, thus allowing the manipulation of such non-geometric, algorithmic relationships as described above. Two-dimensional laser cutting was the primary fabrication method, allowing for rapid manufacturing, and in some cases iterative physical testing. The two technologies have led in the workshop to working methods that extend the geometric schema: the first, by forcing an explicit understanding of design as procedural, and the second by encouraging physical experimentation and optimisation. The resulting projects have tended to focus on responsiveness to conditions either coded or incorporated into experimental loop.

Parametrics

Where standard CAD constructs isolated geometric primitives, parametric models allow the user to set up a hierarchy of relationships, deferring such details as specific dimension and sometimes quantity to a later point. (Kolarevec 2003; Woodbury 2006) Usually these are captured by a geometric schema. Many such relationships in real design however, cannot be defined in terms of geometry alone. Logical operations,

environmental effects such as lighting and air flow, the behaviour of people and the dynamic behaviour of materials are all essential design parameters that require other methods of definition, including the algorithm.

Parametric modelling implies a very different concept of the design. In traditional CAD and drawing, the focus is on the representation of final form described geometrically. This can be easily modified and reworked by the designer. The primitives used in CAD packages (lines, arcs, splines, etc.) are based on drawing conventions evolved over centuries and on physical drawing. Parametric methods, by contrast, force a prior procedural representation in which the form is only the result of this designed procedure. When revising a model it is generally impossible to make changes at the level of final representation (form); it must be modified at the procedural level. The primitives in use still include lines and surfaces, but these are associative rather than simple geometry, and new ones such as global variables and rectangular arrays are added that are based on computation. Relations between elements are constructed prior to instantiation in real space, and so relations dependant on space and scale and the usual fine adjustments to final form are not as easily accommodated.

This is often a cause for frustration in design students introduced to parametric design for the first time. In the education and practice of design, in sketching, exchange and criticism, the final form is the object, which (especially during the earliest phases) can be interpreted many different ways. This is in fact an extremely important part of the creative process either by the individual or in groups. A parametric model instead requires a great deal of explicitness and effort up front to create the schema, and the ability to thereafter switch explicit, procedural representations of a form is not permitted. The advantage of this approach is the assistance it gives to the designer in structuring ones thoughts. To understand this conceptual shift, it has been our position that the skills of the programmer are necessary in the future of design.

Introduction to Generative Components

Bentley's Generative Components (GC), currently in beta testing, was made available to the twenty six Bartlett students, half drawn from the design diploma units and half MSc students in Adaptive Architecture and Computation, with prior programming experience. The GC software is based on a traditional CAD interface, except that a project is simultaneously represented by two models: one geometric and the other procedural. This second, symbolic model is a hierarchical graph that represents each geometric element as a node and the relationships between elements as directed links. Geometry can only be placed in the model in relation to prior geometry: a line, for instance, can be drawn between two prior points, and this then shows up as a lower node in the graph with links showing dependency to the two points. When a geometric element in the graph is updated, all lower elements that are dependent on it are also immediately redrawn to reflect their new position, similar to moving a control point on a spline.

The object orientated nature of the software allows a model, with all its geometry and behaviour to be encapsulated into one single component and used as a single piece of geometry. A single schema for a truss or a glazing panel, for instance, might be repeated throughout the project with variation in shape at every instance controlled by the logic and relationships built into the component. This logic can be stated entirely with dependent geometry, or coded at various levels from simple mathematical expressions through scripting to fully compiled C#. The software thus allows designers with varying levels of computing knowledge to work as deep as they wish, and provides for the model a method of modularity in which each element is flexible and capable of computation.

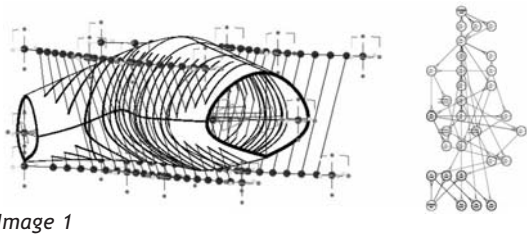

Image 1

Advantages of the generative approach

The obvious advantage of such a method, exploited in most examples of generative design, is that the algorithm can reproduce a schema with changes to its dimension or configuration, and thereby create forms both of great complexity, and with a responsiveness to conditions and environment. A project by Takehiko Iseki typifies this, in which solar paths, mineral content and other environmental factors of a salt water site serve as parameters to form a surface ideally suited to salt crystallisation on the edge of the Dead Sea. The result is a sweeping curved plane that responds to the changes in the site and environment with millions of unique elements. Such a form as the generative model produces is well beyond

the capacity of a human designer to realise through conventional drawing, still less through construction methods based on standardisation.

Image 2

Limits of the approach

But any tool, as with any method of working, influences the result of the designer's creativity. For all the flexibility that a universal programming language provides, the structure of GC allows some forms to be produced with ease while others require rather more effort. Over the past three years of workshops held during the development of the GC software, the most common method for producing geometry has been to replicate a component over the rectangular UV grid of a doubly curved b-spline surface.

While programming languages and geometry are universal in intent, their constraints on the design process were still notable during the workshop. The default data structures of computer languages (in particular the rectangular array) replace one schema limitation with another. The indexing of data in this way is conceptually hard-wired into much of our thinking both in CAD and in code. This is in part because the structure and primitives are based not on geometry but programming logic, on

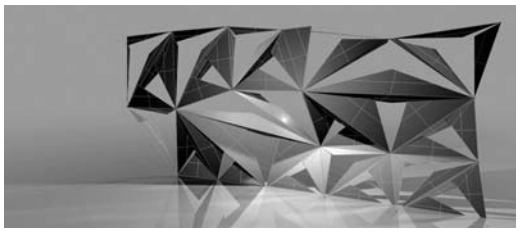

Image 3

such structures as rectangular integer-indexed arrays of data. Even the project above, for all its complexity, is still a repetition of a modular element, based on a rectangular grid.

Thus, approximately half of the projects in the workshop attempted in some fashion to

Image 3b

subvert this grid. A collaborative work between Michael Georgiou and Yiannis Kanakakis involved the placement of triangular components on a topologically irregular surface, the logic of which is immediately grasped in spatial terms, but difficult to define parametrically. Thankfully this can be overcome with a little programming, using an algorithm to sort and index node points, but this requires effort. The number of projects which have required such a step suggests that more intuitive, or spatial methods of data access, might be developed in the future.
While any method of working imposes certain constraints, the direct control one has over geometry in manual drawing or traditional CAD reflects the way we think about form. The potential danger of the generative approach lies in the fact that its procedural structure may be too abstract and distinct from the needs of the project. In the worst case, the designer is guided into making a form that makes sense only virtually.

A way forward

But the solution to this problem also potentially lies in the fact that this method of working is procedural rather than geometric. What is being designed is not the final form, but the steps taken to generate that form, and if these steps are closely grounded in the reality of the project, then the generative process actually forces one to consider the logic of the design on a deeper level than that of simple geometry and the drawing.

This is the advantage of equipping designers with algorithmic thought. The teaching aims of the workshop have been to forge a closer connection between this procedure of form generation and the real, constructed design in one of two ways: either in how it is built or in how it performs. These two areas are particularly aided by the generative, procedural approach because they are themselves time dependent processes, the logic of which is not captured in the traditional, static drawing. In the first case, the method of fabrication may be dependent on tool paths and tolerances that can be simulated in the script file that creates the geometry, so that the designed object is actually built virtually, rather

than simply drawn as a complete form. This forces the designer to consider the underlying construction logic with some rigour even during the initial design phase. In the second case, that of performance, we consider optimisation. The behaviour of a virtual model can be simulated by the algorithm to predict properties such as structural efficiency or mechanical movement, and when coupled with an initial parametric schema the repeated simulation can be used to find a final form that meets stated goals. Due to time constraints, fabrication during the workshop was limited to two-dimensional laser cutting. While GC has inbuilt functions for segmenting and flattening curved surfaces for this largely automated process, the consideration of a logical order of assembly of triangulated or quadrilateral segmented strips was found to be highly useful in planning the initial model.

Rather than exploring optimisation through virtual analyses, the workshop provided an opportunity for some to combine both the automated fabrication process with optimisation. Although virtual simulation was considered for several projects, the parametric setup along with rapid prototyping allows the testing of models to occur in physical reality. Wei Shan Chia's project for a deployable structure made from folded surfaces arrayed a simple hinged module over a complex curve, adjusting the geometry of each instance to fit. The changes in shape as the model is folded could have been simulated, but was constructed to test both the changes in form as the structures fold and the tolerance of the materials. In Fred Guttfield's bird table, components were designed to move in response to the weight of the birds interacting with the project. The balance, in particular, of these was fine tuned by testing many different prototypes made from the parametric model.

Alternatively, a virtual optimisation process can be implemented that takes the fabrication method into account. To manufacture doubly curved surfaces using the laser cutter or from CNC milled tiles, Martha Tsigkari developed an agent simulation to walk the surface, estimating errors and finding an optimal geometry for quadrilateral patches. An arbitrary surface can be used as a guide, and the algorithm set to design to tolerances and limits dictated either by a project brief, or by the machining process to be used in construction.

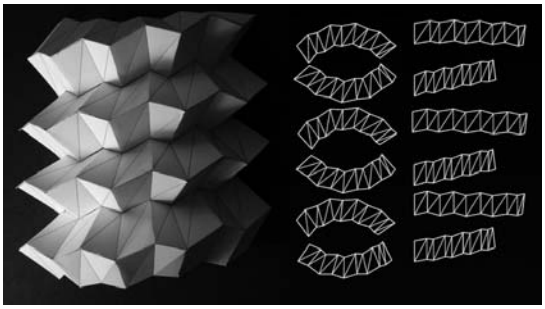

Image 4

Conclusion

Much generative design today is produced for its own sake, simply as an exploration of the method. It is still a new technology and requires new techniques of modelling and thinking.
A great deal of activity as a new user and in workshops such as this is therefore focused on getting over conceptual hurdles such as the formal predisposition of grid, or the hierarchical nature of a dependent model.
We see the design studio of the future as integrating process and manufacturing. Based on what we see currently in development, it seems a closer connection with design and building process is possible by aligning the procedural working method implied by such generative software either with the procedure of fabrication or with optimisation of the final form. In both cases the designer is made to consider aspects of a project's behaviour as a matter of course, rather than simply its geometry. It is our hope and belief that in both cases the designer can have a much closer relationship to the process of making.

Acknowledgements

We would like to thank Robert Aish and Bob Sheil for helping to organise the workshop, and all the participants, particularly Carolina Briones Lazo, Przemek Jaworski, Takehiko Iseki, Michael Georgiou, Yiannis Kanakakis Wei Shan Chia, Fred Guttfield and Martha Tsigkari, who have allowed their work to be published.

References

Kolarevec B (2003): Architecture in the Digital Age: Design and Manufacturing.
Taylor & Francis, London.

Loukissas Y and Sass L (2004): Rulebuilding: a generative approach to modelling architecture using 3D printers. In Beesley P, Cheng NYW and Williamson RS (eds.) proceedings: ACADIA 2004.

Woodbury R (2006): Every Designer is an Editor. In Beesley P, Hirosue S, Ruxton J, Turner C and Trankle M (eds.) Responsive Architectures: Subtle Technologies 2006. Riverside Architectural Press.

Keywords:
Parametric, Generative, Fabrication, Programming.