Introduction

The architectural design process is almost always iterative. Designers create solutions that, in turn, pose new questions, which are then investigated to generate more refined or even entirely new solutions. Designers often use computer-aided tools to build models and help them visualize ideas. However, the vast majority of these models are still built in such a way that they are difficult to modify interactively. The problem becomes more severe when bespoke 3D models are geometrically complex. Changing one aspect of such a model usually requires extensive low-level modifications to many of its other parts. To address this problem, designers have begun using parametric design software, which allows them to specify relationships among various parameters of their design model. The advantage of such an approach is that a designer can then change only a few parameters and the remainder of the model can react and update accordingly. These derivative changes are handled by the software, but are based on associative rules set by the designer. Associative and parametric geometry, in essence, describe the logic and intent of such design proposals rather than just the form of the proposal itself. This kind of design both requires and helps to create powerful interactive tools that allow designers to explore and optimize a multitude of possibilities while reducing the amount of time it takes to do so in a rigorous manner. Engaging these parametric and algorithmic processes requires a fundamental mindset shift from a process of manipulating design representations to that of encoding design intent using systematic logic. Algorithmic thinking calls for a shift of focus from achieving a high fidelity in the representation of the appearance of a design to that of achieving a high fidelity in the representation of its internal logic. The advantage of algorithmic thinking is that it can build '... consistency, structure, coherence, traceability and intelligence into computerized 3D form'.[1] Parametrically and algorithmically built models can react with high fidelity to their real-life counterparts when subjected not only to user changes of geometric parameters, but also to structural forces, material behaviour and thermal and lighting variations, as well as contextual conditions. Because they accurately represent the internal construction logic of the structure at hand, parametric models can also be unfolded or translated into geometries that can be digitally fabricated. This powerful digital workflow of parametric form-finding that is influenced by design intentions as well as performance analysis and digital fabrication logic is one of the defining characteristics of current digital architectural practice. Contemporary architects, such as Patrik Schumacher, partner at Zaha Hadid Architects, have gone as far as coining *parametricism* as the name of a new movement in architecture following *modernism*. He writes: 'We must pursue the parametric design paradigm all the way, penetrating into all corners of the discipline. Systematic, adaptive variation and continuous differentiation (rather than mere variety) concern all architectural design tasks from urbanism to the level of tectonic detail. This implies total fluidity on all scales.'[2] He points out that the fundamental themes in parametric design include versioning, iteration, mass-customization and continuous differentiation. It is helpful to briefly define these terms.

Versioning

Borrowed from the software development field, the term *versioning* refers to the process of creating versions – or variations on a theme, if you will – of a certain design solution based on varying conditions. Parametric software allows the designer to create a prototype solution that, rather than being cast in a static CAD file format, is *wired* – almost as a string puppet would be. This wiring allows the design solution to be tweaked and manipulated, creating new versions when new forces and conditions arise.

Iteration

Again borrowed from the software development field (see a pattern here?), the term iteration refers to cycling through or repeating a set of steps. In the case of parametric architecture, iteration can, in principle, create variation at every pass through the same set of instructions. Examples may include varying the size and shape of a floor plate as one builds a skyscraper, or changing the angle of a modular cladding system as it is tiled over an undulating surface. In addition to producing variation, iteration can be a powerful tool for both optimization and for minimizing the time needed to achieve that optimization. Using a fluid parametric system, which can give immediate feedback, a designer can generate solutions and test them rapidly by iterating through many possibilities, each created with a different set of parameters.

Mass-customization

One of the main successes of the industrial revolution is the idea of mass production. Factories and robots are able to produce thousands of copies of the same prototype. However, given the advent of digital fabrication technologies, we are now able to change the manufacturing instructions between each object. Given that the process

1 Terzidis, K. *Expressive Form: A Conceptual Approach to Computational Design.* Routledge, 2003.

2 Schumacher, P. *Parametricism – A New Global Style for Architecture and Urban Design, AD/Architectural Design – Digital Cities,* Vol. 79, Iss. 4, July/August 2009

is parameterized and robotic, it often costs the same to mass-customize the manufactured products as it does to mass-produce the same quantity of identical products.

Continuous differentiation

Another borrowed term, this time from the field of calculus, continuous differentiation alludes to a feature of versioned, iterative and mass-customized parametric work that allows for difference to occur within a continuous field or rhythm. As opposed to mere variety, parametrically varied instances within an overall group, curve or field maintain their continuity to other instances before and after them while uniquely responding to local conditions.

The characteristics of a parametric design system

The question then becomes, what is a parametric design system, and how can it help improve the design process or more rigorously explore possible design alternatives? In addition to the themes defined above, all parametric design systems share several characteristics and have similar constructs: *object-orientation*, *classes* or *families*, *methods* and, of course, *parameters*. Let us briefly define these concepts.

Object-orientation

Modern parametric software usually uses an object-oriented approach in its design. Object-oriented programming is a well-established computer science topic that is beyond the scope of this book, but a brief description is in order.

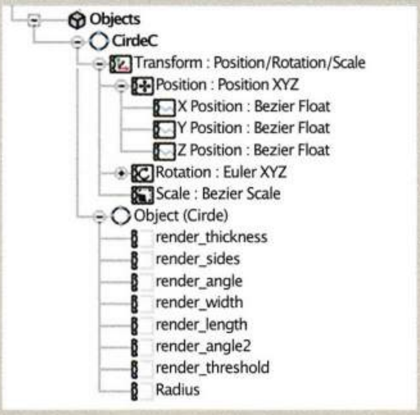
The user interacts with a parametric system in a manner that reflects its internal algorithmic strucure, by creating and modifying *objects* such as circles, spheres, doors and walls. A parametric system usually stores these objects in an object-oriented database that can be accessed, searched and modified.

Each object then has *values* that determine its *attributes*. For example, a circle will almost always contain an attribute called *centre* or *position* and another one called *radius* **[fig. 4]**. It will also probably contain an attribute called *name* that identifies the circle. It is usual for a certain value of an object to be represented with reference to the object and attribute with which it is associated. A popular notation is to use a full stop to separate an attribute from its parent object: *object.attribute*. Thus, if one wishes to reference the value of the radius of a circle named *circleC*, one might encounter the following term: *circleC.radius*. Similarly, the X-axis position of the same circle could be the X attribute of the position attribute of the circle, as in *circleC.Position.X*, and the Y-axis value could logically follow as *circleC.Position.Y*.

Values can either be constants (e.g. 100 m) or functions, which need to be evaluated to compute a final value. The power of a function in the value placeholder of an attribute is that it can derive its value from the values of other attributes, which can belong to other objects. Consider the following hypothetical function of a radius of a circle:

C.Radius = distance(PointA, PointB)

The above expression specifies that the radius of a circle is not a constant number, but is derived from the distance between two points (*PointA* and *PointB*). In such a case, we call the variable *C.Radius* a *dependent* variable as it depends on other values. We also describe such constructs as *associative* – as in *associative geometry*. Association of parameters with one another allows us to derive unknown entities from known ones. In the above example, if the distance between the two points ever changes, the circle's radius will change



```
Objects
  CircleC
    Transform : Position/Rotation/Scale
      Position : Position XYZ
        X Position : Bezier Float
        Y Position : Bezier Float
        Z Position : Bezier Float
      Rotation : Euler XYZ
      Scale : Bezier Scale
    Object (Circle)
      render_thickness
      render_sides
      render_angle
      render_width
      render_length
      render_angle2
      render_threshold
      Radius
```

fig. 4 The attributes of a circle in Autodesk's *3ds Max* software.

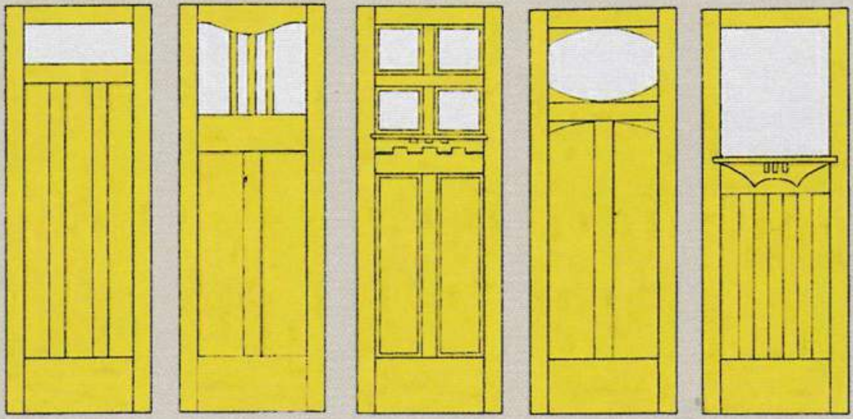


fig. 5 A family of Scottish doors.

accordingly. We call this feature of updating the value of one object based on changes in other values *propagation*. Imagine a large network of wired or associated values. A change in one or few parameters would propagate through the whole network, modifying the values of attributes and changing the characteristics of the final design solution. This is the power of an associative parametric system. Objects, attributes and values are associated with one another and parameterized so that a change in the value of one parameter can have ripple effects throughout the design.

Families and inheritance

Objects that share certain characteristics can be organized as members of a *class* or *family* of objects. A class or family of *doors* **[fig. 5]**, for example, can contain many individual family members (hinged doors, sliding doors, folding doors, etc.). The advantage of grouping several objects into a family is that they can then share certain attributes with their siblings and inherit certain attributes from their parents. It is much more efficient to organize these shared attributes only once, in a parent object, than to have to customize all the attributes and values for each offspring.

Methods

In an object-oriented system, methods are functions and algorithms that act on an object by modifying its attributes. Rather than have a large set of centralized instructions that specify how to draw circles, squares and triangles, an object-oriented system delegates, encapsulating these instructions in the class or family of each object. How an object is to be constructed or modified is thus encoded as a method in the object itself. In the case of a circle, one such method could be to construct the circle by specifying the position of its centre and the value of its radius attribute. Another method could be to specify three points that circumscribe it. The system can simply tell a circle to draw itself – or it can ask a door to reverse its opening. In a modern parametric system a typical object, even one as simple as a sphere, can have many parameters and methods **[fig. 6]**.

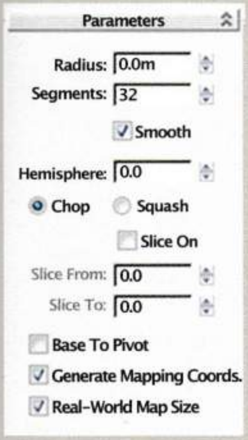
Parameters

At the heart of any modern parametric system is the term *parameter* and so it would be wise to define that term at this point. The word *parameter* derives from the Greek for *para* (besides, before or instead of) + *metron* (measure). If we look at the Greek origin of the word, it becomes clear that the word means a term that stands in for or determines another measure. The word *parameter* is often confused with *variable*, but it is more specific. In mathematics *parameter* is defined as a variable term in a function that 'determines the specific form of the function but not its general nature, as a in $f(x) = ax$, where a determines only the slope of the line described by $f(x)$'.[3]

In parametric CAD software, the term *parameter* usually signifies a variable term in equations that determine other values. A parameter, as opposed to a constant, is characterized by having a range of possible values. One of the most seductive powers of a parametric system is the ability to explore many design variations by modifying the value of a few controlling parameters **[fig. 7]**.

The remainder of this book presents a series of parametric design patterns of increasing complexity followed by exemplar case studies that reflect the potential of the associated patterns. The book ends with a discussion of the future of parametric design and its potential to form a language of design. The afterword by Brian Johnson closes the discussion with advice on how to craft new solutions, based on knowledge gleaned from this book.

3 From http://dictionary.com


Parameters

Radius: 0.0m
Segments: 32
✓ Smooth
Hemisphere: 0.0
● Chop ○ Squash
☐ Slice On
Slice From: 0.0
Slice To: 0.0
☐ Base To Pivot
✓ Generate Mapping Coords.
✓ Real-World Map Size

fig. 6 The parameters and creation method options of a sphere in Autodesk's *3ds Max*.

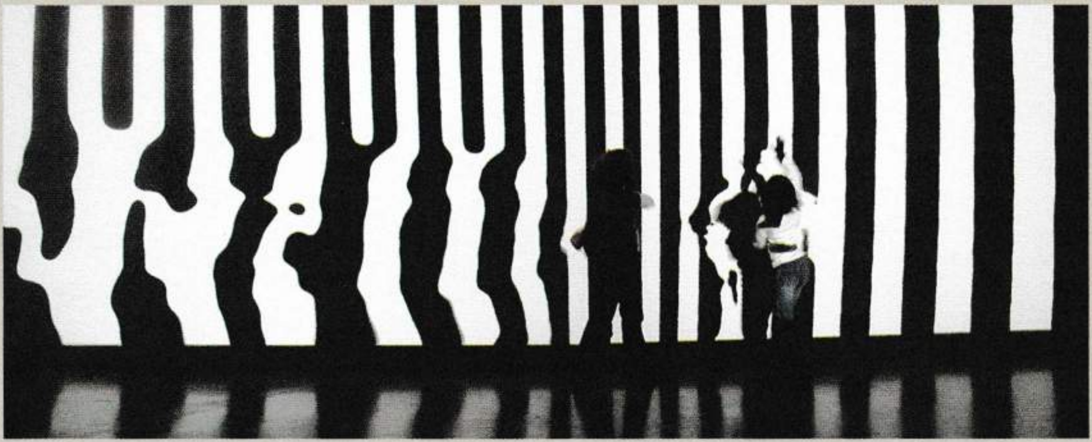


fig. 7 In the *Matière à Rétro-projeter!* (Material Projects) exhibition at the Centre Pompidou, Paris, France, young visitors affect a regular field of light and shadow patterns as they move in front of it.