Chapter 9

Design space exploration

by Mehdi (Roham) Sheikholeslami

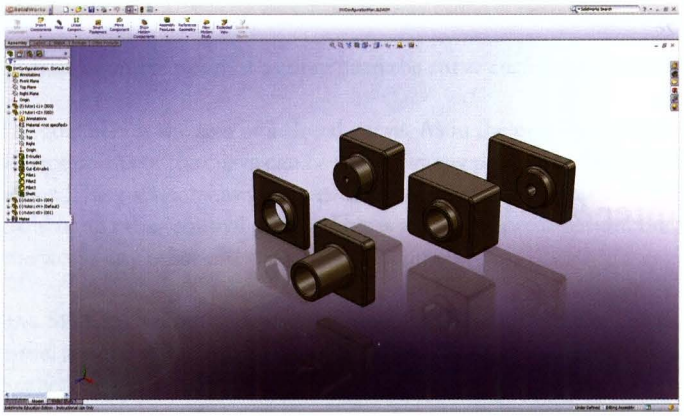
9.1 Introduction

Clearly, exploring multiple alternatives can lead to better designs. Despite this well-known fact, current computer-aided design systems provide only the most rudimentary tools for generating, storing and visualizing alternatives. *Hysterical space* is a novel approach to discover alternatives in the solution space by using the interaction history with a parametric model.

Implicit in any parametric model are the states a designer might have reached by combining variable settings in new ways. Such a model exhibits hysteresis, that is, path dependence – thus the name hysterical space. Based on my Master's thesis (Sheikholeslami, 2009), I present a simple definition of hysterical space as the Cartesian product of variable settings. It provides orderings of the space that yield feasible interactive search strategies. In turn, the orderings suggest interface designs, which I report as working prototypes. Limited user evaluation supports a claim that hysterical space may be a useful approach to design space exploration.

Given our limits, we rely utterly on our external memory to achieve complex tasks (Norman and Dunaeff, 1994). Computation holds out the promise of making this medium active, that is, being able to perform some of the cognition externally.

A key limitation, explained largely by short-term memory capacity and latency, is our ability to create, compare and consider alternatives. Hysterical space is a new concept and computational device for wresting many alternatives from a small number of designer interactions with a parametric modeling system.



9.1: SolidWorks® Configuration Manager is a way to explore multiple variations of a single model.
Source: SolidWorks® screenshots reprinted with the permission of SolidWorks.

Current CAD systems focus mainly on *single states,* with notable exceptions such as the SolidWorks® Configuration Manager (Figure 9.1) and the Autodesk Showcase® (Figure 9.2). Predictably, designers find workarounds – they invent techniques (mostly manual) for design alternatives, such as copying entire files or copying parts of the model within the same file, often using layer structures.

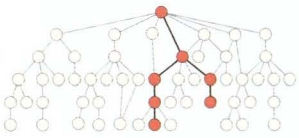
9.2: Autodesk Showcase® has a feature for storing alternatives. Different materials and design are stored as alternatives of a sports car.
Source: Autodesk screen shots reprinted with the permission of Autodesk.

9.1.1 Design space

Despite the differences in design theories, we can claim that all of them admit the existence of a space that contains the solutions to a design problem.

This work mainly uses Woodbury and Burrow's (2006) definition of design space,which, on one hand, describes a general conception of design space and, on the other, entails specific mathematical concepts describing a limited, but sound and tractable design space representation. In short, they argue that design activity is well-modeled by a network structure and the extent of this network is determined by the strategies and the structure of the designer's exploration. They define several terms such as *implicit*, *explicit* and *hysteresis*, which we explain as follows.

Implicit Design Space: This all-encompassing object comprises every possible design solution reachable by a symbol system. It is a network depicting those paths that contain all design solutions, feasible or not, complete or not, that may or may not be visited by the designer.

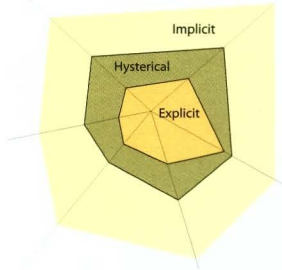


9.3: The explicit space (in orange) expresses the path(s) taken by a designer to reach a solution. It is a subgraph of the implicit space (in grey).

Explicit Design Space: Woodbury and Burrow (2006) argue that an explicit design space comprises those states that have been visited, in the current or an available past exploration episode. They state "design space paths are embedded in both implicit and explicit design spaces". The latter is a smaller portion of design space. Explicit design space is developed through the design process. It gains its structure through the exploration behaviour of designers; especially through choices of strategies reflecting the limits of either computation, designer knowledge; or both.

Design Hysteresis: Woodbury et al. (2000) coin the term *design hysteresis* in the *Erasure in Design Space Exploration*. In essence, the idea is to use data from the explicit space states to construct (discover) implicit nodes by erasing and recombining known data. In this definition, design hysteresis is a part of implicit space of solutions that may not be explicitly visited by the designer during the design process. In fact, design hysteresis discovers explicit states in the implicit space without direct designer action.

We coin the terms *hysterical state* to describe states in design space that are reached by such recombination, and *hysterical space* to describe the set of states so reached. This work poses the research question, "How can a parametric modeling system support the recombination of prior decisions into new states that are meaningful to designers?"



9.4: Graphical representation of implicit, explicit and hysterical space.

9.1.2 Alternatives and variations

We narrow our research to parametric modeling, specifically, to architecture and building engineering. The reasons for this choice are that (a) recently more architects and engineers are using parametric modeling tools in their design; and (b) parameters admit model variations, which enables design space exploration.

A parametric model is an adaptive structure based on a set of parameters. The values of the parameters at any given time define a (usually infinite) space of

model instances. Therefore, parametric models intrinsicly enable exploring great numbers of alternatives and variations.

As a heuristic, we define alternatives as structurally different solutions to a design. In contrast, variations are the design solutions with identical model structure but having different values assigned to the parameters. Typically in a parametric modeling system, variations are considered informally, by moving input points and altering values on an ad hoc basis. Hysterical space expands the set of instances visited to a larger space of variations. We hypothesize that this hysterical space could be a novel approach to enhancing the design process by recombination of prior decisions into new states that are meaningful in design.

9.2 Hysterical space

Defining a hysterical space requires a representation scheme and an interaction history. The representation scheme describes the symbol structures with which we compute a design and its consequent hysterical space. The designer interacts with the representation to create the explicit space – the set of designs actually visited. The interaction history describes what a designer has done and how (s)he has done it – hysterical space amplifies these actions into a collection of representations.

There are many ways to characterize a hysterical space from a parametric model. This work illustrates only the most obvious – the Cartesian product of visited parameter values. The interaction history of the independent variables of a fixed parametric model gives the explicit space, that is, a collection of variations of a parametric model.

To illustrate hysterical space, we developed two design patterns – RECORDER and HYSTERICAL STATE in addition to those in Chapter 8. A RECORDER stores a designer's interaction history with the model, and a HYSTERICAL STATE generates new variations based on the data the RECORDER captures.

9.2.1 Recorder pattern

Currently there is no clear support in the parametric systems for recording the user interactions. Changes to parameter values and other interactions with the system simply flow with minimal recording processes. We introduce the RECORDER pattern that expresses the idea of storing variations of a model based on explicit user choices. Since the current configuration of a parametric model is defined entirely by its parameters, restoring a model to an earlier state requires only reassigning the original values to the parameters (Figure 9.5).

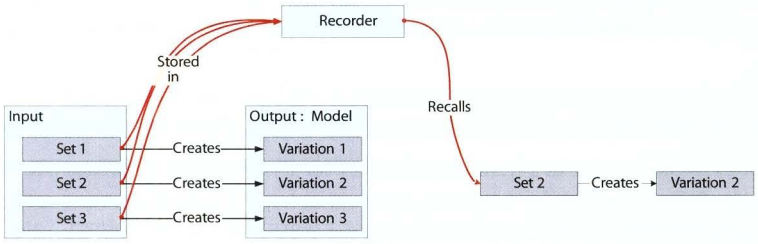
Like the other design patterns in Chapter 8 we define the RECORDER pattern in the following structure:

What. Store user-defined choices selectively.

When. Record some of the explicitly visited variations of a model in order to revisit them in the future.

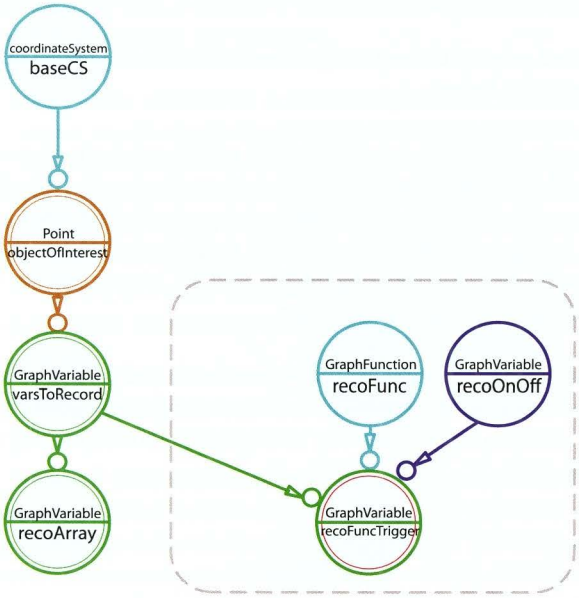
Why. Design is an iterative process in which several variations of one alternative will be visited in the design process. At each phase the designer, the design team or the clients may choose some of these variations for further development. Therefore, having a system that provides the possibility of storing the desired variations seems essential. By using the RECORDER pattern one can restore the design model to a previously visited state.

How. First, identify the variables that define the desired part of the model. Second, create an array storing the recorded values of those variables. Third, restore the model to a desired state by reassigning the corresponding recorded values in the array.



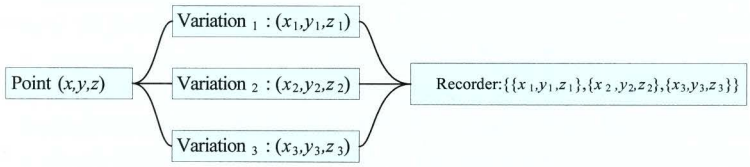
9.5: A schematic view of the RECORDER pattern. Three assignment sets of variables create three different variations of the model, and by recording the second set the designer can restore the model to the second variation.

Our primary tool to assess these ideas is GenerativeComponents®. Figure 9.6 illustrates the symbolic representation of the RECORDER pattern for a simple point. The *object of interest* is the part of the model that the designer desires to record. The parameter *varsToRecord* is the variable or variables from the object of interest that the designer chooses to record in the system. The designer may choose to focus on only a subset of the interaction history. The *recoArray* stores all these recorded values. The nodes of Figure 9.6 inside the dashed rectangle show the mechanism of the recording process in GenerativeComponents®. The *recoOnOff* is a Boolean variable specifying whether to record the values or not. If *true*, the RECORDER will record the values, otherwise not. The function *recoFunc* is the core of the recording process that records values whenever it is triggered. The *recoFuncTrigger* triggers the recorder function. Which object should trigger the function is the decision of the designer. It could be anything in the file that updates the model. Based on the structure of the software we may not need this variable, however; we use this variable to cleanly distinguish the recording process from the model.

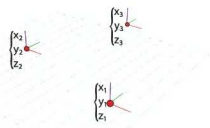


9.6: The structure of the RECORDER pattern in GenerativeComponents®.

For example, a point in a frame is defined by its x-, y- and z-coordinates (see Figure 9.7). Therefore, by recording these variables, the system can restore the position of the point to a desired state (Figure 9.8).



9.8: The recording array for storing the variations of the point contains $x, y,$ and z.



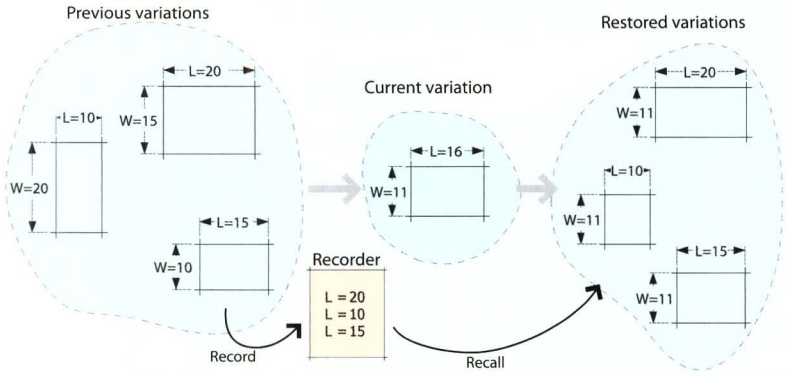
9.7: A point in frame is defined by x, y and z coordinates.

9.2.2 Hysterical State pattern

We limit our examples to the simplest version of the hysterical space, that is, the Cartesian product of the recorded values. The HYSTERICAL STATE pattern illustrates the implementation of the Cartesian product hysterical space.

A parametric model can be restored to an earlier state entirely or partially, based on the parameters that are recorded in the system. If we record all the parameters of the model, we will be able to fully restore it to the recorded state.

On the other hand, recording just some of the parameters gives us the option to restore only parts of the model related to those parameters (Figure 9.9).



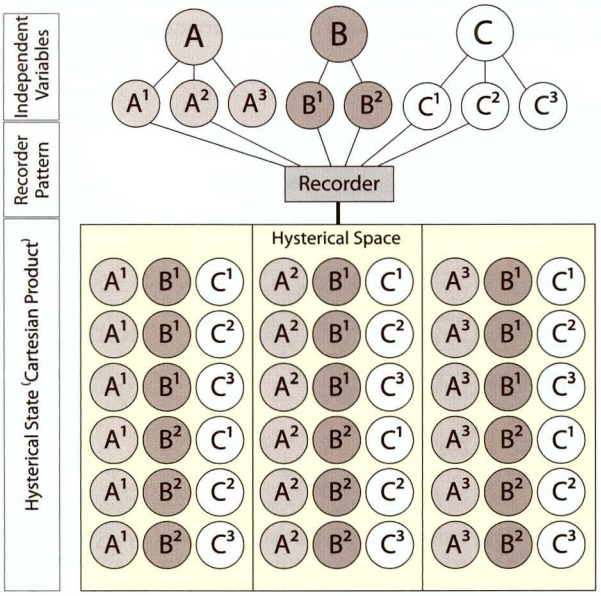
9.9: Recording parts of the model results in partial restoration. By recording only the length of the rectangle, the restored version retains the current width with the stored lengths.

What. Create new variations of a parametric model by recombining prior decisions stored as recorded parameters. In our case, this recombination is the Cartesian product of the recorded values.

When. Explore more variations of a model based on what has been explicitly visited in the previous stages.

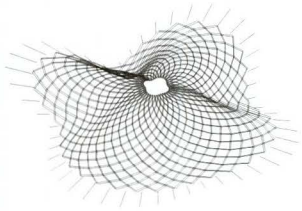
Why. The HYSTERICAL STATE pattern uncovers new nodes in the implicit design space by combining previously recorded parameter values. Visiting the resulting new model variations may lead a designer to explore novel and maybe meaningful directions.

How. The first step is to record the user interactions with the model with the RECORDER pattern. The next step is to generate the Cartesian product of the recorded values. The simplest function iterates through all the recorded values making all combinations. Recreating a model for each of these combinations results in all possible variations in the hysterical space (Figure 9.10).



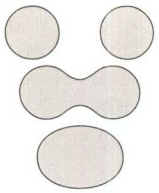
9.10: By generating the Cartesian product of the recorded values, the system can provide several variations of the model.

9.3 Case study



9.11: A roof variation for the Aviation Museum modeled in GenerativeComponents®.

The *Aviation Museum* was the capstone project for my Master of Architecture degree (Sheikholeslami, 2006) (Figure 9.11). One part of the museum is used in this case study of hysterical space. This comprises a single roof corresponding to the size of the objects underneath it. For example, larger aircraft result in a larger roof span. Each primary exhibit object in the museum is represented by a circle, such that the circle encompasses that object (Figure 9.13). To cover these circles with a single roof, I used the Metaballs implicit surface representation. Metaballs are visually organic objects in n-dimensional space (Blinn, 1982). I used a two-dimensional Metaballs algorithm for the museum roof plan.

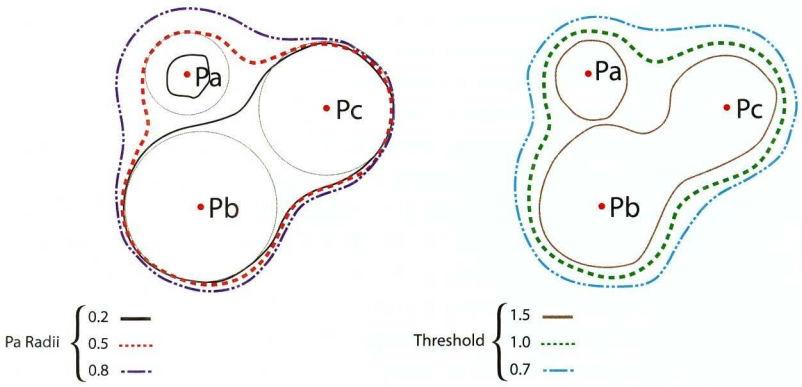


9.12: Two Metaballs distort and combine based on proximity.

Since all of the recorded parameters in this example affect the 2D plan of the roof, in Figure 9.13 we briefly describe the logic relating the plan and roof.

- Create a point grid for the museum base and the Metaballs algorithm. Parameters control both size and density of the grid. The parameter *gridDensity* specifies the density of the point grid and, consequently, the smoothness of the museum's roof.
- Position circles on the point grid to represent the main exhibits in the museum. The radius of each circle (parameter *radii*) reflects the size of an aircraft. (Figure 9.14). We refer to each circle by its centrepoint.

- Apply the Metaballs algorithm to generate the boundary of the roof. The parameter *threshold* defines how much a MetaObject's surface, in our case a circle, influences other MetaObjects. As the threshold increases, so does influence that each MetaObject has on others (Figure 9.14).



9.14: (a) The effect of *radii* of the points on the museum's boundary, in this figure radius of (Pa) changes. (b) The effect of the *threshold* on the museum's boundary.

We record four parameters, three of which are the radii of the circles $\{r_a, r_b, r_c\}$ and one is the *threshold* t (Equation 9.1).

$$\begin{aligned} V &= \{V_1, V_2, V_3, V_4\} \\ &= \{r_a, r_b, r_c, t\} \end{aligned} \tag{9.1}$$

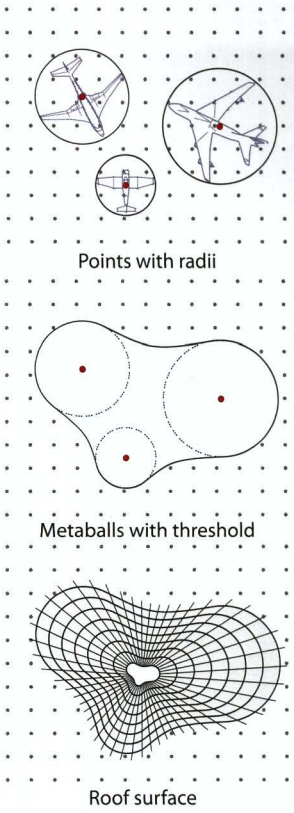
We record two variations of the roof with the following values for the recorded parameters:

$$\begin{aligned} S^0_* &= \{r^0_a, r^0_b, r^0_c, t^0\} \quad \text{(first variation)} \\ &= \{0.5, 1.0, 0.6, 1.0\} \\ S^1_* &= \{r^1_a, r^1_b, r^1_c, t^1\} \quad \text{(second variation)} \\ &= \{0.7, 0.5, 0.8, 0.6\} \end{aligned} \tag{9.2}$$

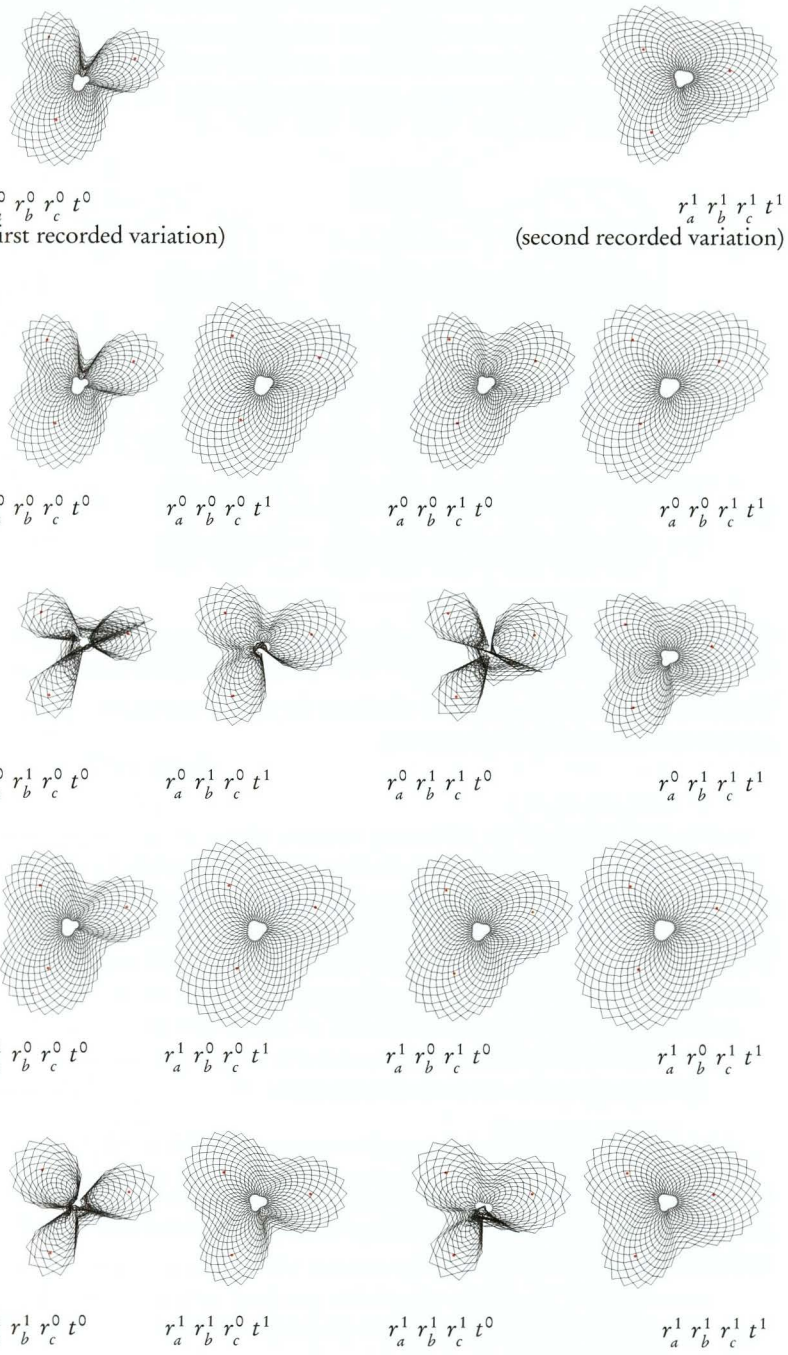
So now we have recorded two values for each of the variables. The Cartesian product of these values generates 16 different combinations and thus 16 roof variations (Equation 9.3).

$$|\mathscr{H}| = |T^*_0| \times |T^*_1| \times |T^*_2| \times |T^*_3| = 2 \times 2 \times 2 \times 2 = 16 \tag{9.3}$$

Figure 9.15 shows the two recorded variations of the roof in comparison with all possible combinations of the recorded values. As you can see, some of those variations are entirely different from what has been explicitly visited.



9.13: The steps to create a 2D plan of the aviation museum.



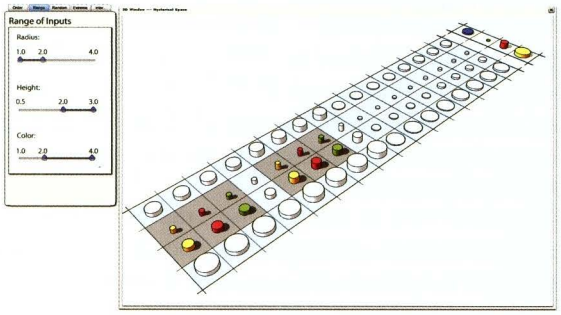
9.15: Sixteen variations of the aviation museum, created by the combining the variables of the two recorded variations.

9.4 Representing the hysterical space

The number of states in a Cartesian product hysterical space (we contract to *hysterical space* when not ambiguous) grows exponentially with the number of variables.

The simplest hysterical space representation comprises lists of its assignment sets, that is, the n sets that from which the Cartesian product is defined. Such a representation is *unevaluated*, which means it must be further processed to produce explicit representation of its states. A naïve evaluated representation is thus an n-dimensional array with each dimension capturing an assignment set. In all but the most simple of hysterical spaces, such a representation would be defeated by sheer size—it would grow exponentially with the number and size of the assignment sets. We therefore seek representations that compute only those parts of the hysterical space actually visited in an interaction, reserving the array representation for those subsets of hysterical space that are rendered in their entirety. A representation can be conceived as a choice of *ordering* a subset of states that are somehow *picked, computed* or *filtered* from the hysterical space. Our strategy will be to use such orderings to pick out subsets of the hysterical space that are then generated and displayed in their entirety as an array.

What, then, is the effective maximum size for a subset of hysterical space that can be represented directly using an array representation? We propose several methods for representing the hysterical space, for example, order of generation, range of inputs (Figure 9.16) and interpolation. See Sheikholeslami (2009).

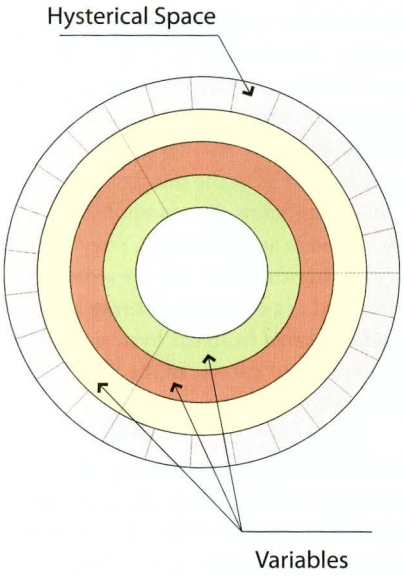


9.16: Representing the hysterical space by a range of inputs. One can filter the hysterical space by specifying the range of the input variables. In this figure the coloured cylinders are the ones that are filtered by the user's choice.

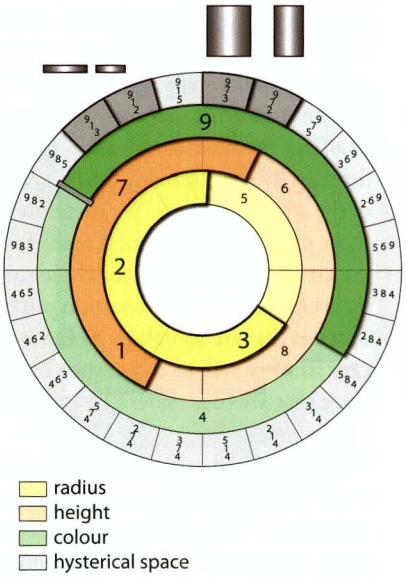
9.5 Visualizing the hysterical space

Since hysterical space is multi-dimensional, visualizing it presents challenges. Although we implemented several prototypes, in this section we describe the main one, called the *Dialer*,. It is implemented in GenerativeComponents®.

The Dialer comprises concentric rings. Each ring represents one parameter and the divisions of a ring correspond to its parameter's recorded values. The outmost ring illustrates the members of the hysterical space. Figure 9.17 shows three variables with three values for each (three division for each of the inner rings). As a result, the $3 \times 3 \times 3 = 27$ divisions on the outmost ring correspond to the items of the hysterical space.



9.17: The Dialer: each ring represents one variable and the outmost ring shows the hysterical space.



9.18: The Dialer for the parametric cylinder, three rings represent three variables and the outmost ring shows the 24 variations of the cylinder.

Each ring has a slider with adjustable size that selects the values on the ring. The Cartesian product of the selected values highlights the corresponding items in the hysterical space (Figure 9.18).

The Dialer in Figure 9.18 shows three parameters – *radius, height* and *colour* – with 3, 4, and 2 values for parameters respectively. By moving and resizing the sliders on each ring, a designer can select the desired values, and as a result, the corresponding items will be highlighted in the outmost ring, which represents the hysterical space. For example, in Figure 9.18, by selecting $\{2,3\}$ for radius, $\{7,1\}$ for height and $\{9\}$ for colour, four items in the hysterical space become highlit.

The circular arrangement of the values in the Dialer makes a relatively more compact visualization than a linear arrangement. However, as the number of values for a parameter increases, the growing number of the divisions in the corresponding ring may defeat this interaction scheme.

9.6 Conclusion

In the domain of parametric modeling, in which models are explicitly defined in terms of a set of parameters, the idea of exploring the designs engendered by the parameter space is already well-established. For all but trivial designs, the space is vast indeed. Hysterical space defines a potentially interesting subspace in which all designs are parametrically close to what has already been found. The Cartesian product model of hysterical space presents a novel way to access these implied states. Its structure is both simple and clear. Its near triviality masks a surprising richness. We were able to quickly envision a variety of ways to order (and thus search and visualize) states in the hysterical space. We are confident that there is much more to discover, in both representation and interaction.

It seems difficult to design visualizations that adequately capture the structure of variations, yet these are important in conveying the mechanism of a model (and thus its implied design space). Furthermore, parametric proximity does not imply geometric similarity of designs. Two models may be very similar in geometry but considerably different in parameters. Conversely, two very close values of the parameters may result in quite a distinct model. Further work would search for algorithms covering a wide range of hysterical space without overwhelming designers by representing very similar variations.

We simply do not understand the cognitive importance of the explicit states, which form the basis for hysterical space. Our interfaces provide no place for these. With the gift of hindsight, we are astonished at our lack of foresight.

We do not know if hysterical space can cause early commitment to a premature design or idea. The Cartesian product hysterical space generates variations of the same alternative, which may not be the best solution to the design problem. The mastermind of the design is still the designer and it is her decisions that lead in particular hysterical space.

This work provides a basis from which to search for new ideas for structuring hysterical space and the promise that there may be some fertile ground to cover in such a search. There may be new discoveries in the generated items that lead to completely new designs. By looking at the variations of hysterical space in the case studies, we discover distinct forms and models from the explicit space that may be worth considering in the design process.

The Cartesian product model *may* be useful in design. This claim gains support from the reactions of designers to the model and interface prototypes, and from our initial (and admittedly idiosyncratic) demonstrations of chosen designs. We informally discussed the idea with a number of designers and applied the hysterical space to their work. From the feedback that we received, it seems that it can be beneficial in their design process. However, more studies with users in real design situations need to be done to determine both positive and negative effects of the hysterical space on the design process.

Contributor biographies

Onur Yüce Gün, BArch (Middle East Technical), MSc (MIT), initiated and led the Computational Geometry Group of Kohn Pedersen Fox Associates New York. Through his work on more than 40 projects from 2006-2009, he focused on developing methods and tools for complex geometric building design. He is currently a faculty member at İstanbul Bilgi University and continues his research and professional practice in his design laboratory: O-CDC.

Brady Peters, BSc (Victoria), BEDS, MArch (Dalhousie), specializes in complex geometry in architectural design and, more recently, in architectural acoustics. First at Buro Happold and then Foster + Partners (where he was promoted to Associate Partner) he worked on projects including the Smithsonian Courtyard Enclosure, the Copenhagen Elephant House, the Khan Shatyr Entertainment Centre, Thomas Deacon Academy, the West Kowloon Great Canopy, and the SECC Arena. Currently, he is a PhD fellow at the Royal Danish Academy of Fine Arts.

Mehdi (Roham) Sheikholeslami, MSc (Shahid Beheshti), MSc (SFU), combines research, teaching and practice in parametric modelling systems. His research is in computational design, parametric modeling and design space exploration.

Bibliography

Aish, R. and Woodbury, R. (2005). Multi-level interaction in parametric design. In Butz, A., Fisher, B., Krüger, A., and Oliver, P., editors, *SmartGraphics, 5th Intl. Symp., SG2005*, LNCS 3638, pages 151–162, Frauenwörth Cloister, Germany. Springer.

Alberti, L. B. and Grayson, C. (1972). *On Painting and On Sculpture. The Latin Texts of De Pictura and De Statua [by] Leon Battista Alberti.* Phaidon. Edited by Cecil Grayson.

Alexander, C. (1979). *The Timeless Way of Building.* Center for Environment Structure Series. Oxford University Press.

Anderson, P. B. (2009). Map projections. Accessed at http://www.csiss.org/map-projections/ on 13 October 2009.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2009). Manifesto for agile software development. Accessed at http://agilemanifesto.org on 29 May 2009.

Berlinski, D. (1999). *The Advent of the Algorithm: The Idea that Rules the World.* Harcourt.

Blinn, J. F. (1982). A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1:235–256.

Borning, A. (1981). The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3:353–387.

Bowyer, A. and Woodwark, J. (1983). *A Programmer's Geometry.* Butterworths.

Bringhurst, R. (2004). *The Elements of Typographic Style.* Hartley & Marks Publishers, 3rd edition.

Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design.* Morgan & Kaufmann.

Carlson, C. (1993). An algebraic approach to the description of design spaces. PhD thesis, Department of Architecture, Carnegie Mellon University.

Carlson, C. and Woodbury, R. (1994). Hands-on exploration of recursive patterns. *Languages of Design*, 2:121–142.

Davies, C. (1859). *Elements of Descriptive Geometry; with Application to Spherical, Perspective, and Isometric Projections, and to Shades and Shadows.* A. S. Barnes and Co.

Dertouzos M. et al., (1992). ISAT Summer Study: Gentle Slope Systems; making computers easier to use. Presented at Woods Hole, MA.

Euclid (1956). *The Thirteen Books of Euclid's Elements, Translated from the Text of Heiberg, with Introd. and Commentary by Sir Thomas L. Heath.* Dover Publications.

Evitts, P. (2000). *A UML Pattern Language.* Macmillan Technical Publishing.

Farin, G. (2002). *Curves and Surfaces for CAGD: A Practical Guide.* Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann.

Flaherty, F. (2009). *The Elements of Story: Field Notes on Nonfiction Writing.* Harper, 1st edition.

Flemming, U. (1986). On the representation and generation of loosely-packed arrangements of rectangles. *Environment and Planning B: Planning and Design*, 13:189–205.

Flemming, U. (1989). More on the representation and generation of loosely packed arrangements of rectangles. *Environment and Planning B: Planning and Design*, 16:327–359.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

Gantt, M. and Nardi, B. A. (1992). Gardeners and gurus: patterns of cooperation among CAD users. In *CHI '92: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 107–117, New York. ACM.

Garrett, J. J. (2002). *The Elements of User Experience: User-Centered Design for the Web.* Peachpit Press.

Gaspard Monge, B. B. (1827). *Géométrie descriptive.* V. Courcier, imprimeur.

Grünbaum, B. and Shephard, G. (1987). *Tilings and Patterns.* W. H. Freeman.

Harada, M. (1997). Discrete/continuous design exploration by direct manipulation. PhD thesis, Carnegie Mellon University.

Henderson, D. W. (1996). *Experiencing Geometry: On Plane and Sphere.* Prentice-Hall Inc.

Highsmith, J. (2002). *Agile Software Development Ecosystems.* Addison-Wesley.

Hoffmann, C. M. and Joan-Arinyo, R. (2005). A brief on constraint solving. *Computer-Aided Design and Application*, 2:655–663.

Itten, J. (1970). *The Elements of Color*. Wiley.

Johnson, W. B. and Ridley, C. R. (2008). *The Elements of Mentoring*. Palgrave Macmillan, revised and updated edition.

Kundu, S. (1988). The equivalence of the subregion representation and the wall representation for a certain class of rectangular dissections. *Communications of the ACM*, 31:752–763.

Lakatos, I. (1991). *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press.

Maleki, M. and Woodbury, R. (2008). Reinterpreting Rasmi domes with geometric constraints: a case of goal-seeking in parametric systems. *International Journal of Architectural Computing*, 6:375–395.

Marques, D. M. (2007). Federation modeler: a tool for engaging change and complexity in design. Master's thesis, School of Interactive Arts and Technology, Simon Fraser University.

Maxwell, R. and Dickman, R. (2007). *The Elements of Persuasion: Use Storytelling to Pitch Better, Sell Faster & Win More Business*. HarperBusiness.

McCullough, M. (1998). *Abstracting Craft: The Practiced Digital Hand*. MIT Press.

Miller, H. W. (1911). *Descriptive Geometry*. The Manual Arts Press.

Mitchell, W. J., Liggett, R. S., and Kvan, T. (1987). *The Art of Computer Graphics Programming: A Structured Introduction for Architects and Designers*. Van Nostrand Reinhold.

Monahan, G. (2000). *Management Decision Making: Spreadsheet Modeling, Analysis, and Application*. Cambridge University Press.

Myers, B., Hudson, S. E., and Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer–Human Interaction*, 7:3–28.

Norman, D. and Dunaeff, T. (1994). *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Basic Books.

Norman, D. A. (1988). *The Psychology of Everyday Things*. Basic Books.

Palladio, A. (1742). *The Architecture of A. Palladio; in four books*. Printed for A. Ward, S. Birt, D. Browne, C. Davis, T. Osborne and A. Millar.

Palladio, A. (1965). *The Four Books of Architecture*. Dover Publications, Inc.

Peters, B. (2007). The Smithsonian courtyard enclosure: a case-study of digital design processes. In *Expanding Bodies: Art • Cities • Environment: Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 74–83, Halifax (Nova Scotia). Riverside Architectural Press and Tuns Press.

Piegl, L. and Tiller, W. (1997). *The NURBS Book*. Springer-Verlag, 2nd edition.

Piela, P., McKelvey, R., and Westerberg, A. (1993). An introduction to the ASCEND modeling system: its language and interactive environment. *Journal of Management Information System*, 9(3):91–121.

Pollio, M. V. (1914). *The Ten Books on Architecture*. Harvard University Press. Translated by Morris Hicky Morgan.

Pollio, V. (2006). *The Ten Books on Architecture*. Project Gutenberg. Accessed at http://www.gutenberg.org/etext/20239 on 11 June 2009.

Pottmann, H., Asperl, A., Hofer, M., and Kilian, A. (2007). *Architectural Geometry*. Bentley Institute Press. Edited by D. Bentley.

Qian, Z., Chen, Y., and Woodbury, R. (2007). Participant observation can discover design patterns in parametric modeling. In *Expanding Bodies: Art • Cities • Environment: Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 230–241, Halifax (Nova Scotia). Riverside Architectural Press and Tuns Press.

Qian, Z. and Woodbury, R. F. (2004). Between reading and authoring: patterns of digital interpretation. *International Journal of Design Computing*, 7. Accessed at http://wwwfaculty.arch.usyd.edu.au/kcdc/ijdc/vol07/articles/-woodbury/index.html on 28 February 2010.

Qian, Z. C. (2004). A pattern approach to support digital interpretation. Master's thesis, School of Interactive Arts and Technology, Simon Fraser University.

Qian, Z. C. (2009). Design patterns: augmenting design practice in parametric CAD systems. PhD thesis, Simon Fraser University.

Qian, Z. C., Chen, Y. V., and Woodbury, R. F. (2008). Developing a simple repository to support authoring learning objects. *International Journal of Advanced Media and Communication*, 2:154–173.

Ramsay, C. and Sleeper, H., editors (2007a). *Architectural Graphics Standards*. American Institute of Architects, 11th edition.

Ramsay, C. and Sleeper, H., editors (2007b). *Architectural Graphics Standards*. American Institute of Architects, 4.0 CD-ROM edition.

Rockwood, A. and Chambers, P. (1996). *Interactive Curves and Surfaces: A Multimedia Tutorial on CAGD*. Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers.

Rogers, D. (2000). *An Introduction to NURBS: With Historical Perspective.* Morgan Kaufmann Publishers.

Rogers, D. F. and Adams, J. A. (1976). *Mathematical Elements for Computer Graphics.* McGraw Hill Book Company.

Rottenberg, A. T. and Winchell, D. H. (2008). *Elements of Argument: A Text and Reader.* Bedford/St. Martin's Press, 9th edition.

Ruhlman, M. (2007). *The Elements of Cooking: Translating the Chef's Craft for Every Kitchen.* Scribner's.

Ruskin, J. (1844). *The Seven Lamps of Architecture:lectures on architecture and painting ; the study of architecture.* A.L.Burt, New York.

Ruskin, J. (1857). *The Elements of Drawing in Three Letters to Beginners.* Smith, Elder, London, 2nd ed. edition.

Sannella, M., Maloney, J., Freeman-Benson, B. N., and Borning, A. (1993). Multi-way versus one-way constraints in user interfaces: experience with the Delta Blue algorithm. *Software - Practice and Experience*, 23:529–566.

Schneider, P. L. and Eberly, D. H. (2003). *Geometric Tools for Computer Graphics.* Morgan Kaufman Publishers.

Schön, D. (1983). *The Reflective Practitioner: How Professionals Think in Action.* Basic Books.

Sheikholeslami, M. (2006). The aviation museum. Master of Architecture Thesis, Shahid Beheshti University.

Sheikholeslami, M. (2009). You can get more than you make. Master's thesis, School of Interactive Arts and Technology, Simon Fraser University.

Smith, T. M. and Smith, R. L. (2008). *Elements of Ecology.* Benjamin Cummings, 7th edition.

Steele, G. L. (1980). The definition and implementation of a computer programming language based on constraints. PhD thesis, MIT.

Strunk, W. and White, E. B. (1959). *The Elements of Style / by William Strunk; with revisions, an introduction and a new chapter on writing by E.B. White.* Macmillan.

Sussman, G. and Steele, G. (1980). CONSTRAINTS - a language for expressing almost hierarchical descriptions. *Artificial Intelligence*, 14(1):1–39.

Sutherland, I. (1963). Sketchpad: a Man–Machine Graphical Communication System. Technical Report 296, MIT Lincoln Lab.

Tidwell, J. (2005). *Designing Interfaces: Patterns for Effective Interaction Design.* O'Reilly Media, Inc.

Tufte, E. R. (1986). *The Visual Display of Quantitative Information*. Graphics Press.

Tufte, E. R. (1990). *Envisioning Information*. Graphics Press.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press. 4th printing with revisions.

van Duyne, D. K., Landay, J. A., and Hong, J. I. (2002). *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison-Wesley Professional.

Vince, J. (2005). *Geometry for Computer Graphics: Formulae, Examples & Proofs*. Springer-Verlag.

Wang, T.-H. and Krishnamurti, R. (2010). Design patterns for parametric modeling in Grasshopper. Accessed at http://www.andrew.cmu.edu/org/tsunghsw-design on 6 March 2010.

Week, D. (2002). *The Culture Driven Workplace*. Assai Pty Ltd.

Weisstein, E. (2009). Wolfram MathWorld. Accessed at http://mathworld.wolfram.com on 7 December 2009.

Williams, R. (1972). *Natural Structure*. Eudaemon Press.

Williams, R. (1995). *The PC is Not a Typewriter*. Peachpit Press, 1st edition.

Williams, R. (2003). *The Mac is Not a Typewriter*. Peachpit Press, 2nd edition.

Williams, R. (2008). *The Non-Designer's Design Book*. Peachpit Press, 3rd edition.

Woodbury, R., Datta, S., and Burrow, A. (2000). Erasure in design space exploration. In *Artificial Intelligence in Design 2000*, pages 521–544, Worcester, Massachusetts. Key Centre for Design Computing, Kluwer Academic.

Woodbury, R., Kilian, A., and Aish, R. (2007). Some patterns for parametric modeling. In *Expanding Bodies: Art • Cities • Environment: Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture*, pages 222–229, Halifax (Nova Scotia). Riverside Architectural Press and Tuns Press.

Woodbury, R. F. (1993). Grammatical hermeneutics. *Architectural Science Review*, 36:53–64.

Woodbury, R. F. and Burrow, A. L. (2006). Whither design space? *AIEDAM, Special Issue on Design Spaces: The Explicit Representation of Spaces of Alternatives*, 20:63–82.

Yorck (2002). The Yorck Project: 10.000 meisterwerke der malerei. DVD-ROM, Directmedia Publishing, GmbH. ISBN 3936122202.

Trademark notices

Adobe Creative Suite® is a registered trademark of Adobe Systems Incorporated.

ArchiCAD® is a registered trademark of GRAPHISOFT.

Autodesk®, AutoCAD®, Maya®, Revit® and Autodesk Showcase® are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries.

Cinema4D® is a registered trademark of Maxon Computer.

form•Z® is a registered trademark of AutoDesSys Incorporated.

GenerativeComponents® is a registered trademark of Bentley Systems Incorporated.

Maple™ is a trademark of Waterloo Maple Incorporated.

Mathematica® is a registered trademark of Wolfram Research Incorporated.

Microsoft Excel® and Microsoft Word® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Rhinoceros® and Grasshopper™ are a registered trademarks or trademarks of Robert McNeel and Associates.

CATIA®, SolidWorks® and Virtools® are registered trademarks of Dassault Systèmes.

Index

Academic Quadrangle, SFU, 267
Adobe, 65
Aish, Robert, 8, 43
Alberti, Leon Batista, 81
Albion Riverside, 35
Alexander, Christopher, 47, 186, 187
algorithm, 12–14, 52, 84, 85, 215, 237
  complexity, 16
  deBoor, **156**, 159–161
  deCasteljau, **148**, 156, 159
  display, **15**, 16
  Metaballs, 282, 283
  ordering, **15**
  propagation, **15**, 16, 25
  recursion, 39, 260–268
  update, 19, 20, 22, 54, 62, 63
angle, 85, 103, 109, 126, 180, 192, 200, 238, 250
ArchiCAD, 65
ASCEND, 12
AutoCAD, 65
Autodesk Showcase, 276

blossom, **157**, 158–160
Brancacci Chapel, 82
Brunelleschi, Filippo, 81
Buxton, Bill, 35

camel casing, 50, 59
Camera degli Sposi, Palazzo Ducale, 83
Cartesian product, **59**, 60, 275, 280, 281, 283, 285–287
CATIA, 65, 186
Chomsky, Noam, 186
Church of Santa Maria Novella, 83
Cinema4D, 65
circle, 20–22, 71, 173, 177, 194, 195, 198–200, 204, 206, 208, 226, 227, 231, 232, 235, 238, 264, 274, 282, 283
  arc, 71
class, **56**, 56–57, 59, 61
  inheritance, 56
collection, 59, 60
conic section, **135**, 166, 228
  cone, 195, 209
  cylinder, 196
  ellipse, 211
  parabola, 214
  torus, 69, 71–73, 173
constraint expression, **13**, 14, 16, 21, 52
convex hull, **149**, 163, 237
Cook, Peter, 219
coordinate system, *see* frame
coordinates
  Cartesian, 20, 192, 208, 212
  cylindrical, 20, 208, 212
  spherical, 212, 216
CSCEC Tower, 177
Currey, Charles, 136
curve, 85, 134–145, 180–182, 193, 203, 204, 208–212, 217, 226, 228, 232, 235, 240, 241, 248, 249, 268, 271, 273, 274
  C continuity, **144**, 144–145, 154, 155
  G continuity, **144**, 144–145
  approximation, 136, **137**
  B-Spline, 152, **155**, 156, 158–167, 172, 204, 206
    knot vector, **156**, 158–162, 164
  Bézier, 42, 146, **146**, 147–156, 159–163, 165–167
  binormal vector, **141**
  composite, 173, 174
  continuity, **144**, 161
  control
    local, **153**, 163, 164
    pseudo-local, **155**
  control point, **137**, 146, 147, 155

control polygon, **137**, 146, 149, 160, 162, 163, 166
curvature, 141, 142
cusp, 144, 145
degree, 143, **149**, 150, 162, 203
evolute, 141
free-form, 135
Frenet frame, **141**, 141–144
hodograph, 143–145
inflection point, 141, 142, 144
interpolation, **137**
normal vector, 139, **140**, 141
NURB, 152, 166, **166**, 167, 172
order, 143, **149**, 155, 156, 162, 163, 203
osculating circle, 141
osculating plane, 141
parametric, **138**, 139
polynomial, 135
tangent vector, **139**, 140, 143
torsion, 142
unit tangent vector, **139**, 140, 141
CustomObjects, 43

data structure, 57–58, 61
array, **57**, 58, 215, 247, 279, 280, 285
list, **57**, 58, 265, 285
network, 57
tree, **57**, 262
dataflow, *see* propagation
DaVinci, Leonardo, 33
Delta Blue, 12
derivative
first, 139, 143
second, 141, 143, 155
descriptive geometry, 22, 82
design space exploration, 64, 277
alternative, 176, 275, 278, 279
alternatives, 275
design hysteresis, 277
explicit design space, 277, 278
hysterical space, 275, 277, 278, 280–282, 285–287
implicit design space, 277, 281
variation, 174, 276, 278–281, 283, 287
Dialer, 285, 286
distance, 85, **94**
signed, **94**
divide-and-conquer, 27–28
dot notation, 13, **13**, 19, 21, 57, 59
dot product, *see* vector, scalar product

Eden Project, 43–45
Emacs, 65
end-user programming, 65–67
Erickson, Arthur, 267
Euclid, 10, 81, 86
expression, 51–54, 62

F3–F5 Towers, 177
Firth of Forth bridge, 220
form•Z, 65
Foster + Partners, 34, 46, 69–79
Fournier, Fournier, 219
frame, 18, 20, 95, 101, 105, **106**, 106–122, 139, 169, 205, 212, 219, 234, 240, 251, 260
composition, 120–122
right-hand rule, **107**, 108, 250
rotation, 117–118, 244
scaling, 118, 244
shearing, 119–120
translation, 120
function, **53**, 53–54, 56, 61, 62, 64, 180, 181, 214–216, 231, 233, 236, 246, 247, 249–259, 279, 281
argument, 53, 54
domain, 252–259
range, 252–259
recursive, 260–268

Gaudí, Antoni, 33, 34
GenerativeComponents, 9, 43, 65, 174, 180, 186, 279, 280, 282, 285
GCScript, 182
geometric construction, 21–22
golden ratio, 33, 52
Gram-Schmit orthonormalization, 110
graph, **13**, 15, 16, 59, 64
acyclic, 12, 15, 17
chain, **13**, 15, 21, 25
cyclic, 13, **13**, 14, 17, 18
directed, 13, **13**, 15
directed acyclic, **15**, 57
link, 13, **13**, 16–18, 192, 202
path, **13**

Grasshopper, 65

helix, 208–210
Hesselgren, Lars, 43
hyperboloid, 199

I_EMS system, 43
index, 60
instance, 59
International Finn Dinghy, 135–137
International Terminal Waterloo, 43–44
intersection, 20, 84, 123–133

Jeanneret, Charles, 33, 226
Jordan Curve Theorem, 249

Kilian, Axel, 8
Kohn Pedersen Fox Associates, 171–183
Kunsthaus Graz, 219

Lamboley, Gilbert, 136
LaTeX, 4, 10
Le Corbusier, *see* Jeanneret, Charles
line, 20–22, 60, 83, 98–103, 123–126, 129, 130, 132, 134, 141, 143, 146, 177, 193, 194, 197, 199, 220, 225, 228, 233, 251, 262, 268, 274
  2D, 98–102
  3D, 103
  collinear, 127
  explicit equation, 98
  implicit equation, 98–99
  normal-point equation, 100
  operator, 99–100
  parallel, 126, 130, 132
  parametric, 138
  parametric equation, 101, 126, 128, 138, 147–151, 158, 162
  perpendicular, 132
  point-vector equation, 131
  segment, 17, 21, 30, 130, 132, 149, 249
  skew, 132, 133
linear interpolation, **138**

Mantegna, Andrea, 83
Maple, 42
mapping, 194
Masaccio, 81–83
Masolino de Panicale, 81, 82
Mathematica, 42
Maya, 65
McLuhan, Marshall, 36
method, **57**
Microsoft, 9, 65
module, **45**, 54, 61, 63
Monastery of Sainte-Marie de La Tourette, 226
Monge, Gaspard, 82
monomial, **149**

name, **13**, 29, 53, 56, 190, 202
Nanjing South Station, 178–183
Nasir-Al-Molk Mosque, Iran, 33
New Elephant House, 69
Nicholas Grimshaw & Partners, 43–45
node, **13**, 16, 19–21, 25, 50, 54, 59, 63, 64
  condense, 16, 18
  dependent, 24, 59
  independent, 24
  internal, 14, 24
  multi-property, 16–18
  predecessor, **13**, 14, 15, 21
  property, 13, 52
  single-property, 14, 16–18
  sink, 14, 202
  source, 14, 24
  successor, **13**, 14–16, 19
  typed, 19

object, **56**, 57
order, 285
  topological, 15, **15**, 16
  total, 15

Palladio, Andrea, 33
parallelogram, 108, 221
parametric design, 15
pattern, 8–10, 185–274
  CLEAR NAMES, 187, 190
  CONTROLLER, 187, 191–200, 206, 237, 251
  GOAL SEEKER, 187, 269–271, 273, 274
  HYSTERICAL STATE, 278, 280, 281
  INCREMENT, 187, 207, 209
  JIG, 187, 201–206
  MAPPING, 187, 252, 254, 257
  PLACE HOLDER, 46, 187, 218–222

POINT COLLECTION, 187, 212–215, 219
PROJECTION, 187, 223–225
PROJECTOR, 241
REACTOR, 187, 230, 231, 234
RECORDER, 278–281
RECURSION, 187, 260, 261
REPORTER, 187, 236–239, 241–244
SELECTOR, 187, 245–251
TREE SORT, 75
perspective, 81, 82, 84
plane, 71, **103**, 103–106, 123, 125–131, 134, 140, 141, 199
implicit equation, 103, 110, 125, 126, 131
normal, 132
normal vector, 103
normal-point equation, 104, 131
operator, 104–105, 110, 125, 127, 129, 130
parallel, 128, 131
parametric equation, 105, 138
point, 17–22, 60, 85, **86**, 86–88, 92, 95, 98, 101, 104, 106, 110, 112, 113, 115, 123–125, 128, 129, 131, 138–140, 177, 193, 194, 197, 200, 208–217, 219, 225, 226, 228, 229, 231–235, 239, 243, 246, 248, 250, 251, 273
co-planar, 126
collinear, 21, 124, 127
intersection, 130
parametric, 139, 146, 168
projection, 102, 105–106, 133, 223–229, 248
polygon, 218, 222, 228, 242, 261, 267
convex, 149
polynomial, **149**
program, 54, 60, 62, 64, 66, 78
bug, 61
control statement, 52–53, 215, 271
flow of control, 52
statement, 51–53
programmer, 9, 37, 49, 53, 55, 57, 60–62, 66, 124
programming, 9, 21, 49–67, 72
language, 49, 52, 54, 62, 66
meta, 64
propagation, **12**, 16–17, 21, 24, 62–64
property, **13**, 14–21, 56, 57, 246
graph-dependent, 14, 16, 19
graph-independent, 14, 16, 19
node-dependent, 19, 20
node-independent, 20
value, 13, 21

quadrilateral, 205, 222

Rasmi dome, 33, 222
rectangle, 254
golden, 265
square, 266
replication, 59, **59**, 217, 234
Revit, 65
Rhinoceros, 65, 172, 177, 186
Rochester House, 35
rotate, 76, 126

Salisbury Cathedral, 33
scale, 76
schema, **13**
sketch, **35**, 35–37, 73
Sketchpad, 7, 11
Smithsonian Institution, 38
snapping, 83, 84
SolidWorks, 65, 186, 276
spiral, 208–211, 213, 260
spline, **154**, 155, 162
spreadsheet, 11, 62
Strunk, William, 10
Strutt, James W., 34–35
surface, **168**, 168–170, 193, 203, 205, 206, 212, 217, 219, 225, 226, 228, 242
curvature, 170, 219
isocurve, 169
normal, 219
principal directions, 170, 219
ruled, 199
unit normal, 169
Sutherland, Ivan, 7, 11
symmetry, 221
systolic array, **138**, 147, 148, 150, 151, 159

tangent, 20–22, 64, 85, 173, 205, 271, 274
Temple Expiatori de la Sagrada Família, 34
ThingLab, 12
TikZ/PGF, 9–10

topological sort, *see* order, topological
torus
    patch, 71, 172, 174
transformation, *see* frame
    mapping, 252–259
triangle, 198, 206, 243, 244
tweening, *see* linear interpolation
type, 19, 50, **54**, 54–55, 57, 59

update method, *see* algorithm, update

value, 50, 51, 56
variable, **50**, 50–55, 59, 73, 246
    global, 54
vector, 21, 61, 85, **87**, 87–98, 103–106, 110, 112–114, 116, 123, 129, 219, 233
    addition, 89, **89**, 90–92, 253
    basis, 93, **93**, 106, 109, 111, 112, 120
    bound, **95**, 234
    converse projection, 97–98, 133
    cross product, **108**, 108–110, 124, 126–128, 131–133, 141, 250
    direction, 94, **94**, 125, 129, 131, 222
    field, 234
    free, **95**
    inverse, **90**, 108
    linear combination, 92, **93**
    linear independence, **93**, 108, 123
    natural basis, **93**
    norm, **94**
    normal, 85, 125, 128, 129
    orthonormal basis, **109**
    projection, 96–97
    scalar multiplication, 89, **89**, 90–92, 253
    scalar product, **95**, 95–96, 103, 126, 250
    zero, **89**, 124, 139
vector length, *see* vector, norm
Villa Rotunda, 34
Virtools, 65
Vitruvian Man, 33
Vitruvius, Marcus Pollio, 33

White Magnolia Tower, 172–178

Pushed by practices wanting and needing to produce novelty, computer-aided design systems are increasingly parametric – that is, they represent designs that change with their input data. People find parametric computer-aided design systems powerful, but hard to learn and difficult to use. Using these new systems well requires knowledge of design, geometry, computing and the structure of parametric systems themselves.

What new knowledge and skill do designers need to master the parametric? How can they learn and use it? That is what this book is about – it helps designers realize the potential of the parameter in their work. It combines the basic ideas of parametric systems with equally basic ideas from both geometry and computer programming. It uses design patterns as its main tool. A pattern is a generic solution to a shared problem. Using patterns to think and work will help designers master the new complexity imposed on them by parametric modeling.

This book explains how to think, model and conceive complex parametric designs. Through design patterns and many examples, it shows designers how to lift their knowledge and skill out of the CAD toolbox into higher levels of design thinking and action.

Robert Woodbury is a professor in the School of Interactive Arts and Technology at Simon Fraser University in Vancouver, Canada. His research is on how people develop and use interactive systems. Through work in computational design, people-centred systems for sustainability and visual analytics, he aims to discover general concepts and designs for systems that people find engaging and useful. He is a former Olympian and current dabbler in sailing.

ARCHITECTURE/DESIGN

Visit the companion website for *Elements of Parametric Design* at
www.routledge.com/textbooks/9780415779876

an **informa** business

ISBN 978-0-415-77987-6

Routledge
Taylor & Francis Group
www.routledge.com

Bentley
Institute Press