Teaching Generative Design

Thomas Fischer
Design Technology Research Centre, School of Design
The Hong Kong Polytechnic University, Hong Kong.
sdtom@polyu.edu.hk

Christiane M. Herr
MA (cand), Department of Architecture
The Hong Kong University, Hong Kong.
candyhk@hkusua.hku.hk

Abstract

Generative design, which integrates multidisciplinary types of expertise in unconventional ways, was reserved just until recently to experienced and highly autodidactic designers. However, growing recognition of the importance of generative design methodologies have resulted in a need to introduce theories and applications of generative design to undergraduate students as part of their design studies. This emerging educational field of *generative design teaching* currently lacks methodologies, teaching experience and introductory study material. Available textbooks related to algorithmic form generation, discussing *algorithmic growth*, *artificial life*, *fractal images*, *emergent behaviour* and the like have originated in the field of mathematics. This resource provides an abundance of examples and generative approaches but when adapted to design education, it poses great interdisciplinary challenges which are addressed in this paper. Experiences in generative design teaching are presented, focusing on the relation between algorithmic reproduction of nature (as emphasized by authors in the mathematical field) and innovation (as commonly emphasized in design education). This discussion leads to a derivation of pedagogic suggestions as early steps on the way towards theories and curricula of generative design teaching, addressed to curriculum planners, generative design teachers as well as novices of the field such as undergraduate students.

1. Introduction

The production of "*generations"* from initial blueprints is immanent to the variance and reproduction of all life. It is quite an obvious idea to adopt this natural approach to human-made design and to realize product generations designers can choose 'fit survivors' from, which promise to make particular sense in given contexts. In this way, generative design represents the design discipline's interest to apply natural inspiration not only in terms of the

creation of products but also in terms of the *process of creation*. This interest has a long history. One early example of generative design thinking Mitchell identifies are Aristotle's musings on the generation of design variations[1].

Though generative design is not restricted to the application of particular types of tools, digital computers have turned out to be specially appropriate for the following reasons: To generate design implies a somewhat industrial approach to production insofar as efficient automation is required to output large quantities of solutions. A programmable universal machine is certainly a very helpful tool in this respect. In contrast to industrial manufacturing however, generative design leaves the monotony of production up to the computer and at the same time overcomes and avoids the monotony of products. Moreover, a significant part of generative design labour comprises permutation of design elements and attributes, which is most easily accomplished by means of symbolic computation. This symbolic representation that is inherent to computer-aided design (CAD) also seamlessly integrates elements of design simulation. Generative design solutions come into existence in form of digital representations, allowing early evaluations before their actual (e.g. physical) modelling, production or application. In this respect, generative design differs vitally from its natural inspiration, which experiments, generates and extinguishes designs in the most blind and unscrupulous ways. Computer-aided generative design (and CAD in general) is also easily integrated with common office, data processing and communication procedures and equipment. As a result, generative design has good reasons to utilize mathematics, programming and computers and often involves *digital toolmaking*.

Today, *genetic algorithm*, *cellular automaton* or *shape grammar* are important and very common keywords in international discourses on CA(A)D but due to their relative novelty in design and their complexity, these approaches are largely neglected in undergraduate design studies. This is not only due to the interdisciplinary involvement of generative design work. Very little has been done so far to develop methodologies, materials and curricula for generative design teaching and to clarify terms and techniques for teaching purposes.

[1] Mitchell, William J. [9], p.29

2. Terminology and Explanatory Models

Two simple reasons for the common lack of generative design education are that a) there is very little introductory material and b) that generative design terminology is still based on rather vague notions. Very frequently, generative design approaches are not explained clearly but illustrated by naming underlying programming paradigms, which are obscure to outsiders and novices. Such blurry unfocussed models can be useful in design teaching to challenge students, to make them curious, inspire them or to incite student research. Nevertheless, when it comes to implementation issues and tough questions, clear terms and concepts are essential. With the following proposals for explanatory models we intend to fill this gap:

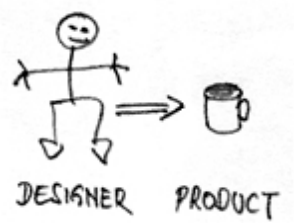

Figure 1: Traditional design approach

Generative design is a design methodology that differs from other design approaches insofar that during the design process the designer does not interact with materials and products in a direct ("hands-on") way but via a generative system.

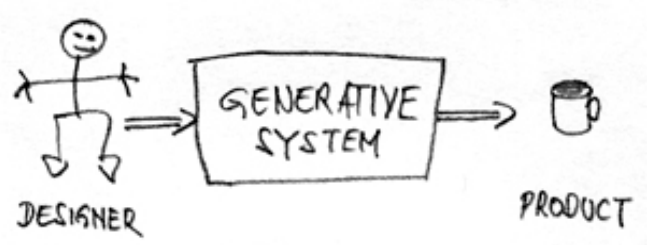

Figure 2: Generative design approach

A generative system is a set-up based on abstract definitions of possible design variations capable of displaying or producing design products (or elements of design products). There is in principle no reason to restrict this approach to the application of digital tools. Fully analogue systems are possible, too. But as digital generative design is of particular interest (see reasons above), this paper mainly focuses on generative CAD.

Computers are ultimately nothing more than symbol processing machines and just like any piece of software, *digital generative design tools are symbol processors*. Generative (symbol

processing) programs of this kind characteristically perform two (explicitly or implicitly distinct) types of operations: The first type generates sets of symbols and the other type "interprets" these symbols by mapping or projecting them onto elements and attributes of design products, thus implementing a manifestation of a possible design. Of each type, there can be as few as one element deployed in a generative unit but more are also possible. Arrays of generative units can run in parallel (e.g. cellular automata). The semiotic relationship between symbol production and symbol interpretation can be anywhere from strict, intentional and meaningful (rational generative design approach) to random (irrational generative design approach).

To explain these terms, we deliberately prefer the term 'explanatory model' instead of 'definition' as it is not possible to draw clear lines between generative and non-generative design. The integration of natural growth and DNA interpretation into design, e.g. by showing timber grain patterns on furniture surfaces might well be considered 'generative'. Another example is the medieval history of gothic building. Lacking means to experiment with physical of mathematical models, medieval builders had to depend on empirical knowledge. This expertise was collected from success as well as from failure of experimental building advances. In this sense, the structural progress of gothic cathedrals represents an early de-facto 'evolutionary' architecture.

3. Potentials, Promises and Myths

Generative design is typically experienced and presented as a very powerful design methodology. Such presentations often imply promises and postulations that are not necessarily entirely true in every case. The following are brief discussions of true potentials and factoids intended to clear up common questions.

One promise that is indeed true is (as mentioned above) that generative systems can generate entire design families or *generations*. The abstraction level at which design solutions are expressed in generative systems guarantees generic capabilities within given (well-defined) problem domains. This allows exhaustive permutation and modification of defined design elements, attributes and parameters and automatic mass generation of possible design solutions.

Generative design is also supposed to enhance the designer's creativity, allowing richer explorations of design spaces. This second promise is typical in its vagueness as it depends on the term *creativity* which itself is not easy to define. As generally known, computers are pretty dumb and only perform what they are programmed to perform, so the idea that generative software can support a creative process appears questionable at first glance. However, the automatic permutation of large numbers of design elements can indeed inspire ideas and concepts, which designers would not necessarily have considered without the support of a generative tool.

A third supposition states that generative systems can be capable of selecting *good designs* from generated designs. This is true in principle but only possible within extremely strict problem domain definitions. In the majority of cases this is not realistic. Computers are powerful tools for creating design variance. But reducing design variance according to criteria of usefulness and beauty needs a great deal of knowledge and common sense. This common sense cannot be put into software easily. Hence, in actual generative design projects, selections from design generations are typically performed by humans.

While the development of generative design systems usually requires programming skills, their application can be comparatively user-friendly and easy for non-programmers. In this sense, a fourth supposition is that generative design tools have sufficient generic qualities to be easily passed on to other designers (with or without programming skills) who need design tools while working on other, maybe similar problems. As generative design tools can output huge design families, this appears to be a particularly promising assumption in regard to design productivity. Though it is of course easily possible to pass a given generative design tool on to other users, doing so does not necessarily embrace the *nature of design*. Due to the uniqueness of every design problem, a generative design tool developed in one design context is not very likely to make equal sense in other design contexts. This matches the authors' observation that designers who develop generative design tools do this quite enthusiastically but designers who are offered the use of other designer's generative tools often respond with refusal. Moreover, a successful generative tool has itself a product-nature insofar as it is its designer's key to generating revenue, which might be a good reason to restrict others from using it.

4. What needs to be taught?

Teaching generative design deals with technique. It is about *how* to generate as opposed to *what* to generate. While experienced generative designers select and modify generative methods according to specific projects, teaching generative techniques initially requires the introduction of a generative toolbox. This toolbox contains mathematical techniques, which in early teaching stages should be introduced in breadth rather than in depth. The open list of emerging areas (toolbox) for generative CAD curricula contains:

- Emergent systems, self-organization (image, sound, animation, behaviour and form) (e.g. cellular automata, swarm modelling)
- Generative grammars (e.g. L-systems, shape grammars)
- Algorithmic generation and growth (image, sound, animation and form) (e.g. fractals, re-writing rules, parametric design, data mapping)
- Algorithmic (re-) production (evolutionary design) (e.g. genetic algorithms, selective procedures)

However, in these fields, design-oriented textbooks (or other types of introductory material) are lacking at the moment. Whereas the way the above techniques are commonly presented implies a strongly reproductive perspective, the key challenge in design is to use them to innovate. This can be supported by emphasizing other generative techniques, which should also appear on this open list such as:

- Data mapping as a symbol-generation technique (e.g. on-line 'data mining') and
- Parametric design as a symbol-interpretation technique

Moreover, supporting and reflective skills should be covered by generative design curricula such as:

- Generative programming (e.g. development tools, languages, AI techniques) and
- Generative aesthetics (e.g. abstraction, symbolic expression, interpretation, generative rhetoric, recognisability, repeatability, accidents and elements of chance, integration of generative design in traditional designer/client/user context)

Classic generative design methodologies such as space-filling curves, genetic algorithms, fractals and emergent behavioural systems have their roots in the realm of mathematics or have at least advanced to canonical exercises in that discipline. Art and design increasingly make use of this instrument. The power these methodologies offer to computer-aided design resulted in new interdisciplinary bonds between design and mathematics. In the design field, this new approach is, so far, mainly being adopted in advanced research and design projects only. Until recently, computer-aided generative student design projects were typically based on extra-curricular learning efforts and in many contexts they are still an exception. Now, the recently growing importance of generative techniques in design increasingly requires a broad curricular coverage in undergraduate design teaching.

However, some pedagogic pitfalls result from the interdisciplinary origin of generative techniques. We argue that these pitfalls are mostly inflicted by design's focus on *open-ended* problems and mathematics' tendency to *close* (or to *tame*) problems.

5. Models of Nature are not Nature

Throughout history, civilizations have developed arithmetic and mathematics and today we are still striving for their further advancing. Discounting base motives, related to warfare and economics, a primary goal of this development was and is the *creation of tools to explain nature*. While mathematics allows strict (algorithmic) formalizations of natural and artificial phenomena, it does not allow for its own validation by its own means (as *Gödel* states in his *Incompleteness Theorem*[2]). Being unable to prove its own truth and still being under development, the history of mathematics must be seen as an open-ended, innovative process and can in this sense itself be described as *design*.

While committed to explaining nature in terms of *true* and *false*, mathematics is not able to prove its own truth by its own means. At the beginning of the 20th century, this finding induces a major setback for formal sciences (and the deterministic world view in general), whose ideal goal it was before to devise a universal formula, or a set of axioms from which all existing phenomena could be deduced. Providing generative (algorithmic, geometric, grammatical etc.) techniques, mathematics finds itself in the ironical position, on the one hand

[2] Gödel, Kurt [8]

not to be able to *ultimately prove statements about nature* but on the other hand to be able to *generate naturalistic designs*.

In order to illustrate the potential of generative calculus, there are two basic areas available: the *natural* and the *artificial*. Illustrations of generative mathematics are strangely attracted to make use of natural examples like clouds, mountains, snowflakes, galaxies, plants and so on. Moreover, basic paradigms for generative strategies are inspired by or borrowed from nature: DNA, evolution, breeding, growing. There are of course good reasons for choosing natural examples for the application of generative mathematics. One is that these are very well known examples and thus good vehicles for explanation of complex mathematical concepts. Another reason for generating naturalistically is the application in the field of virtual reality production, which spends great effort to advance to more and more naturalistic outputs.

However, when it comes to educational material such as student textbooks, the distinction between explanatory model, chaotic surprise and intentional design goals becomes imprecise. The inevitable 'fractal landscapes images' (see figure 3) which can nowadays be mass-generated from specialized stand-alone programs for example, are typical examples of this confusion. In these generated landscapes, parameters and algorithms, geometries and colour schemes are intentionally tweaked to generate even more realistic landscapes including rock textures, trees, reflections on water surfaces and snowy mountain peaks.

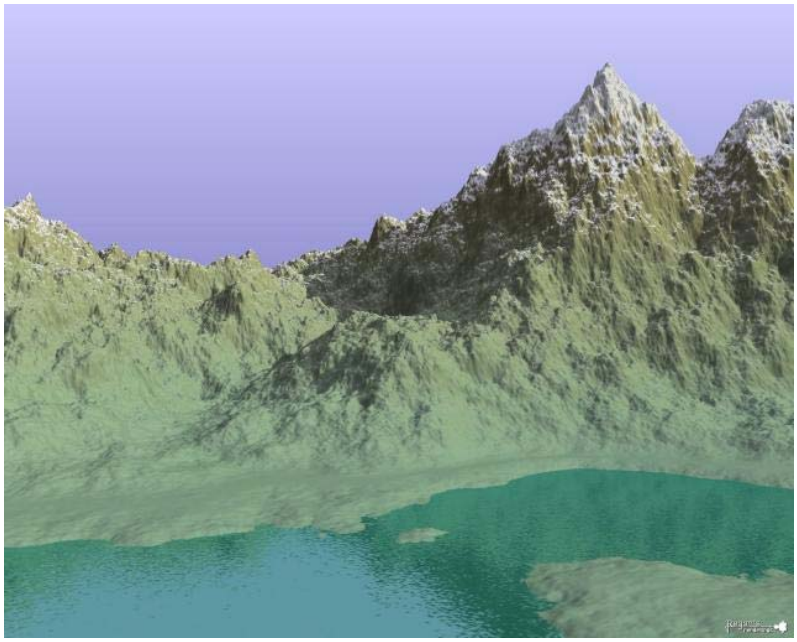

Figure 3: Fractal landscape[3]

[3] Courtesy Roger B. J. Baron, http://meta-x.org/~regor/F-Render/

From an external perspective (e.g. from the view point of design students), this and other generative typologies excessively obscure the concepts they are based on.

It is not immediately obvious that with mathematics, fractal images based on a toolset which in itself is aesthetically passive and neutral, are generated by systems which have intentionally been set up to produce naturalistic outputs. In fact, fractal images like the above (and their underlying algorithms and parameters) are intentionally adjusted to generate natural output in goal-driven and therefore somewhat *alchemic* processes.

On the contrary, images like this which often come along with elaborations on chaos theory, `extreme mathematical monsters'[4] and the like, suggest some sort of *deeper truth* and meaning in mathematics.

Figure 4: Non-self-similar fern, presumably not by Barnsley [5]

A similar example is an image that appears in Bovill, supposedly showing *Barnsley's fern* (see figure 4). The fern has become a self-similar (fractal) classic amongst naturalistic illustrations of generative output, Bovill presents this non-self-similar and supposedly even more naturalistic image, citing Peitgen et al. who point out that:

"The importance of Barnsley's fern to the development of the subject [feedback and iteration] is that his image looks like real fern, but it lies in the same mathematical category as the gasket, the Koch curve, and the Cantor set. [The category of iterated function systems] not only contains extreme mathematical monsters which seem very distant from nature, but it also

[4] Bovill, Carl [2], p. 53
[5] Reprinted from Peitgen, Jurgens and Saupe in Bovill, Carl [2], p. 52

includes structures which are related to natural formations and which are obtained by only slight modifications of the monsters."[6]

This "monstrous" rhetoric is used to explain how iterative function systems (or replacement systems) like the Koch curve can be adapted to generate more complex, more organic and more naturalistic output. However, the shown image does neither look like anything generated by a replacement system because (in contrast to the fern Barnsley originally presented) this one is not self-similar. Though it does not look like natural fern either, its irregular and organic structure suggests to be particularly naturalistic. This "super-natural" output indicates to the layman that mathematics would bear a higher truth from which nature itself might have been generated and that this truth is now encapsulated inside the generative software that has put it out in a *surprising*, perhaps even *mysterious* way.

In this sense, descriptions of generative techniques tend to present mathematics not as a system to develop models for understanding nature but as the cause of nature itself. Spitefully, one could wonder if this tendency is a compensation of the incompleteness of mathematics: "If mathematics is not sufficient to find and prove universal laws behind nature and to explain a snowy mountain or a plant, let's use mathematics to generate some naturalistic mountains or plants from fractal algorithms and show that there might as well be a (universal) mathematical formula behind it!" As mentioned above, Mitchell mentions Aristotle as an early generative design thinker. What he does not mention is that Aristotle is also the originator of mimesis, the adoration of nature by its imitative representation, which obviously has a latent presence in generative mathematics and culminates in the idea of *artificial life*.

It is not the responsibility of mathematics to develop products; mathematics develops tools. When mathematicians develop generative techniques and explore their potential, this happens in a rather playful way. However, it must be stated that in effect, the common selection of naturalistic illustrative themes transports a message, which has negative consequences (not only) for generative design teaching.

After mathematics has been developed to provide models for explaining the world, this logic is inverted when (intentional) output naturalism is now used to 'prove' mathematics. One risk mathematics in general and generative design in particular are therefore facing, is to fall back into assumptions which were common before twentieth-century physics cleared up the

[6] Bovill [2], p.52

previously confused relationship between nature and models. The fact that a model is not identical with what it represents must not be obscured. It would be ridiculous to assume a real building would catch fire because a model of that building is set on fire. But when it comes to mathematical models and computer software that attempts to behave in naturalistic ways, we tend to do exactly that: the model is easily mistaken for the real thing.

6. Generative Teaching of Generative Design

Unnoticed by the generative design field, a pedagogic theory of the same name, "Generative Learning" has been proposed and discussed in the educational field since 1974[7]. First introduced by Wittrock, this approach no longer considers learning as a passive reception of information but as an activity. It is thus following the reform-pedagogical tradition and the constructivist view of learning. The essential contribution of this theory is to state that learners actively organize and transform presented information according to their individual expectations, to the information's relevance from their point of view and to prior knowledge. This perspective appears highly appropriate for a field of study which has an obvious need for providing a solid base of knowledge and techniques which then have to be claimed, be interpreted and transcended in innovative ways.

Generative design and generative learning have more in common than just their adjectives and we argue that the latter is a highly appropriate choice for teaching the former. Both fields are like-minded and easily connected with constructivist thinking, teaching and design teaching. This reflects for instance in the School of Design's Interactive Systems Design[8] stream, in which generative design techniques are taught using turtle robots and (a haptic flavour[9] of) the programming language *Logo* which were both developed by Papert[10] and Minsky, following Piagetian constructivist tradition. As constructivist learning theory, generative design and generative learning put special emphasis on processes, individual approaches to progress and development, tools and tool development. We recommend the pursuit of this line of thought when future design curricula and courses integrating generative approach are laid out.

---

[7] Wittrock, Merlin C. [12]
[8] see the School of Design's *Interactive Systems Design* homepage at http://i.sd.polyu.edu.hk
[9] Fischer, Thomas, Cristiano Ceccato and John Frazer [5]
[10] Papert, Seymour [10]

7. Conclusion

Despite the tendency of design teaching to focus increasingly on interdisciplinary, cultural and conceptual issues rather than being concerned with particular techniques and skill requirements, generative design and its growing significance in the design field require in-depth exercises and studies of techniques, technologies and methodologies. To a certain extent, this constrains generative design teaching to more traditional bottom-up approaches in which, at least in initial stages of learning, skills are prioritised over application. This can partly be compensated by asking students to develop non-computerized generative systems in early stages of learning, requiring no technical skills but merely a basic understanding of generative design. For teaching generic skills, a possible canon of contents with strong roots in the field of mathematics was presented above. The critical issue is that the present need for generative design teaching is not satisfied at the level of this (reproductively oriented) canon. Following design's imperative to innovate and to challenge, the skills acquired when examining basic generative techniques need to be applied and transcended in actual design projects. Generative learning provides a highly suitable paradigm for setting up learning situations accordingly. Once those projects have been developed, students' toolmaking can be subject to critical reflection and the question "*What can generative design do, what can it not do*?" Ultimately, answers to these questions can only be developed not by producing surprising imitations of nature but by innovating generative designs, as Wittrock states: "generation, not discovery is the process of comprehension."[11]

8. Acknowledgements

The authors gratefully acknowledge the support from academic and research staff of the School of Design, the Interactive Systems Design stream at the School of Design, in particular Julian Gibb, Assistant Prof. Catherine Hu, Prof. John Frazer and the Design Technology Research Centre of The Hong Kong Polytechnic University. We also thank Roger "Regor" B. J. Baron for courteously providing *'Fractal Landscape 280394'*, generated with his software "*F-Render*".

[11] Wittrock, Merlin C. [13], p. 353

References

[1] Mandelbrot, Benoit B. [1983]: The Fractal Geometry of Nature. Freeman, NY.

[2] Bovill, Carl [1996]: *Fractal Geometry in Architecture and Nature*. Birkhaeuser, Boston MA.

[3] Ceccato, Cristiano [1999]: *Microgenesis. The Architect as Toolmaker: Computer-Based Generative Design Tools and Methods*. In: Soddu, Celestino (ed.): The Proceedings of the First International Generative Art Conference. Generative Design Lab at DiAP, Politecnico di Milano University

[4] Dawkins, Richard [1987]: *The Blind Watchmaker. Why the Evidence of Evolution Reveals a Universe Without Design*. Norton, NY.

[5] Fischer, Thomas, Cristiano Ceccato and John Frazer [2001]: *Haptic Programming with Machine-Readable Models*. In: Mark Burry et al. (ed.): The Proceedings of Mathematics and Design 2001. The Third International Conference, School of Architecture and Building, Deakin University Australia.

[6] Flake, Gary W. [1998]: *The computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaption*. MIT Press, Cambridge Mass.

[7] Frazer, J. [1995]: *An Evolutionary Architecture*, Architectural Association, London, UK.

[8] Gödel, Kurt [1931]: Über formal unentscheidbare Sätze der *Principa Mathematica* und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38 pp. 173-198

[9] Mitchell, William J. [1977]: *Computer-Aided Architectural Design*. Van Nostrand Reinhold, NY.

[10] Papert, Seymour [1980]: *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, N.Y.

[11] Schmitt, Gerhard [1993]: *Architectura et Machina. Computer Aided Architectural Design und Virtuelle Architektur*. Vierweg, Braunschweig

[12] Wittrock, Merlin C. [1974]: Learning as a generative process. *Journal of Educational Psychology*, 67, pp. 484-489

[13] Wittrock, Merlin C. [1990]: Generative processes of comprehension. *Educational Psychologist*, 24, pp. 345-376