The Value of Digital Simulation in Design Conception

Josh Lobel

Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA, USA

Abstract. Novel insights into the nature of architectural design problems can be derived from digital simulations. By transforming the subject of design interrogation from a mental construct into a numerical one, causal assumptions that may go unnoticed and unappreciated in an implicit design process are required to be made explicit as conditional statements. Assuming there are indefinitely many ways to associate design intentions with design manifestations, relational decisions necessarily made in the process of code-writing shed light on the implied, or desired outcome of a particular design solution.

1 Introduction

The greatest value of simulation in design is not as a method of visualization, but one of interrogation. The construction of a digital simulation[1] requires aspects of a design problem to be abstracted into terms that can be operated on mathematically through logic-based methods. Because logic-based systems inherently rely on the relational hierarchies of the elements within the system, those aspects of the design description which are inherently considered most important by the designer will naturally become the primary elements of the system from which all subsequent elements derive their meaning and value. This method establishes an externalized construct of the specific forces which motivate (drive) and resist (constrain) design solutions.

Simulations can be thought of as "exercisable diagrams"[2] which utilize interactive and behavioral platforms. While such systems are by definition deterministic[2], they are often considered unpredictable because the number of conditional rules they contain create a vast number of possible rule combinations, greatly reducing the observable predictability of the system. Such simulations provide a method for the dynamic evaluation of a given rule set. Designers can ignore rules when it suits them to do so; however, a programmed construct cannot violate the conditional statements upon which they are formed, making them a valuable tool for the analysis and exploration of design intent.

The project entitled *theOffice* is an abstract office environment in which autonomous 'workers' establish their own dynamic organizational hierarchy. When introduced to the system, each worker is tasked with an assignment which requires them to collaborate with a minimum of two other workers based on randomly assigned skill-sets. The goal of *theOffice* was to determine the potential of using goal-oriented simulations to provide feedback on design intentions - which could otherwise not be anticipated from more traditional, static forms of representation - during the conceptual design phase, rather than creating specific design solutions.

---

1 In this context digital simulation is taken as a closed-loop, dynamic, rule-based system created by encoding multiple discrete conditional procedures which are employed recursively and in a combinatorial manner resulting in discernable macro, or global system behavior.

2 Any closed-loop system has a finite set of rules, indicating that it is possible to determine all possible combinations of those rules.

The first part of the paper discusses the importance of intentionality in design, positioning the designer as an active instigator rather than a passive observer, and touches on important differences between static and dynamic methods of design representation. Next the role of simulation is presented as a critical design tool based upon the fundamentally different ways in which design ideas are expressed in comparison to other forms of conceptual representation. Section 2.3 places the investigation in the context of other methodological approaches to simulation within the field of architecture. Section 3.0 details the approach and structure of *theOffice* simulation as a construct. Results and discussion follow in sections 4 and 5, and the paper concludes with thoughts on the implications of this project and future potential for simulation within the broader architectural community.

2 Design Interrogation

Many aspects of a design project are likely change throughout the design process including: scope, program, budget. Within a rule-based system, a particular design manifestation can be iterated to accommodate such change without sacrificing the clarity or rigor of the underlying logics upon which it is built. In contrast, purely formal designs tend to require significant reconsideration as the project develops. By codifying an explicit set of rules, designers can create a rigorous conceptual framework within which the merits of various formal alternatives can be analyzed.

2.1 Advantages to Multi-Variant Approaches

Translation is reinvention [2]. When translating between two or more systems of conceptual representation, hidden or non-obvious discontinuities between the systems become apparent by way of transposition and overlap. Such discontinuities often indicate a gap within a design concept, and can be filled to create a more robust design description. By varying their methods of representation, attention can be redirected to the design concept rather than any particular design representation. The idea is to establish a process of repeated analysis from different view points by treating different methods of representation as design interrogators. An example in architectural design is the use of graphical images along with physical design models. Image-based representation, typically visualizations of plans, sections, elevations, or perspectives of a project, allow designers to isolate various spatial relationships and moments by transforming 3-dimensional space to 2-dimensional space. The only constraints of an image are based upon the process used to create the image, be that material or digital. As Ivan Sutherland stated in 1975, “Pen and ink or pencil and paper have no inherent structure. They only make dirty marks on paper…The behavior of a computer-produced drawing, on the other hand, is critically dependent upon the topological and geometric structure built up in the computer memory as a result of drawing operations[7].” Sutherland touches on a key issue with regard to the critical nature of representation based upon the underlying structure necessary for its creation. Tangible, or physical, representations carry with them all the inherent constraints and limitations of the physical environment as well as the particular material from which they are made. As no two physical objects can occupy the same space at the same time, we learn at a very young age that all physical materials possess an inherent collision detection system. Since most graphical representations have no such limitations, the study of a design through both physical and graphical means is likely to resolve a greater number of non-obvious issues, assuming the representations share a high degree of fidelity.

The key to this approach is providing of a means for comparative analysis, a method for gauging the efficacy of a design through various iterations. The goal is not to arrive at one solution, but to arrive at many solutions. In *Sciences of the Artificial*, Herbert Simon argues that solving a problem through a particular form of representation is an act of making evident that which was already inherent, albeit obscure, in the presentation of the problem [6]; a reference to the notion of problem-setting versus problem-solving [4]. Simon goes on to discuss the critical importance of representation in the design process[3] as well as the importance of continued development in theories of representational methodology[4]. Marvin Minsky refers to a similar methodology for multi-variant representation as "reformulation" [3], or using alternative conceptual descriptions – contexts – to 'see' differently, better enabling the seer to draw potentially non-obvious conclusions about a given entity.

2.2 How Simulation Differs from Other Forms of Conceptual Representation

The speed and ease with which a representation can be altered, the ability to predict the behavior or outcome of the representation, and the ability to anticipate the effects of such change on the representation are factors which can distinguish various modes of conceptual representation. In architectural design, these modes take the form of static images (orthographic projections, perspectives) and physical models. These methods require a specific degree of intentionality in which every aspect of the representation must be actively crafted by the designer. Such methods employ top-down approach to representation, meaning that the outcome is anticipated in advance, and all subsequent work is judged on the ability to reach that goal. These methods can be achieved both "by hand" and through digital means.

Modifications to a static, or fixed, representation are always local changes, they do not automatically propagate throughout the entire system. This is because the relationship of the constituent elements to the overall system is implied and indirect. Further, the time necessary to update or iterate fixed imagery is directly proportional to the extent of the desired variation.

Relative representations require the a priori establishment of a rule set upon which the system can be built, creating an explicit cause-and-effect relationship among elements. Any change to the system results a 'ripple' effect where discrete, localized modifications propagate throughout the system along association chains. Although highly deterministic, the predictability of a relativistic, or parametric system may be non-obvious based on the logic used to create the model. Manipulating the values of parameterized elements is easy; however, altering the topology or fundamental logic of such a model requires redefinition the systems logic. Parametric systems are similar to fixed images in their top-down approach to achieving a presupposed outcome. They differ in that they require a greater amount of time to be spent initializing, or defining the relationships of the system, before any specific output can be created.

Simulations are unique in that they introduce the element of time to relational models. In a computer-programmed simulation, discrete aspects of a referent system are abstracted into a logical, conditional construct. By abstracting the referent system

---

3 "…a deeper understanding of how representations are created and how they contribute to the solution of problems will become an essential component in the future theory of design." [6]

4 "…even though our classification is incomplete, we are beginning to build a theory of the properties of these representations. The growing theories of computer architectures and programming languages – for example, the work on functional languages and object-oriented languages – illustrate some of the directions that a theory of representations can take." [6]

into a series explicit rules, discrete variables can be manipulated and their individual effect on the system studied. The combinatorial and recursive application of rules over time provide simulation a level of self-determination, making them highly unpredictable, albeit still deterministic, systems. These provisions create the potential for concept interrogation and comparative analysis among a large number of design iterations is a short amount of time.

Simulation in the architectural design process is becoming increasingly common as design solvers. The Groningen Stadsbalkon is a pedestrian plaza which is held aloft by 150 non-uniform, inclined columns, that create a space below which can accommodate 3000 bicycles. To determine the location, size, and incline of each column, geometry consultants *designtoproduction* created a simulation which allowed the architects (Kees Christiaanse Architects & Planners) to study various iterations by controlling the placement of individual columns along with a set of parameters that were used to define the properties of the columns without obstructing established walking and cycling paths [5]. The difference between the approach used at Groningen and that which is addressed in this paper is the use of programmed constructs a method of formal inquiry (solver-method) versus one of conceptual inquiry (interrogation-method).

3 *theOffice* Simulation

The construct of *theOffice* uses object-oriented programming to create an abstract workforce which relies on an explicit rule-set to interact with its members and their environment. The simulation itself is produced when the rules are conditionally applied to the workforce over time.

During the simulation the workers (depicted as colored circles) interact in an abstract office environment, increasing and decreasing their size, or sphere of influence, (depicted as a circular red halo about each worker) as a result of their proximal location to fellow workers (Fig. 1). Every worker will attempt to increase their influence as much as possible within the office, however they are required to make room for all members of the workforce without overlapping each other. Each worker carries a set of data which includes: their own unique worker number (these are assigned based on the order in which the workers are created), an assigned pace at which they initially move about the office, a personal rolodex which can store information on up to five other workers, and a particular skill which is represented as a randomly assigned integer. Each worker keeps a record of the number of times they come into contact with other workers, storing information of their five most recent contacts in their rolodex. These contacts are graphically indicated by lines drawn from each worker to their rolodex-contacts.

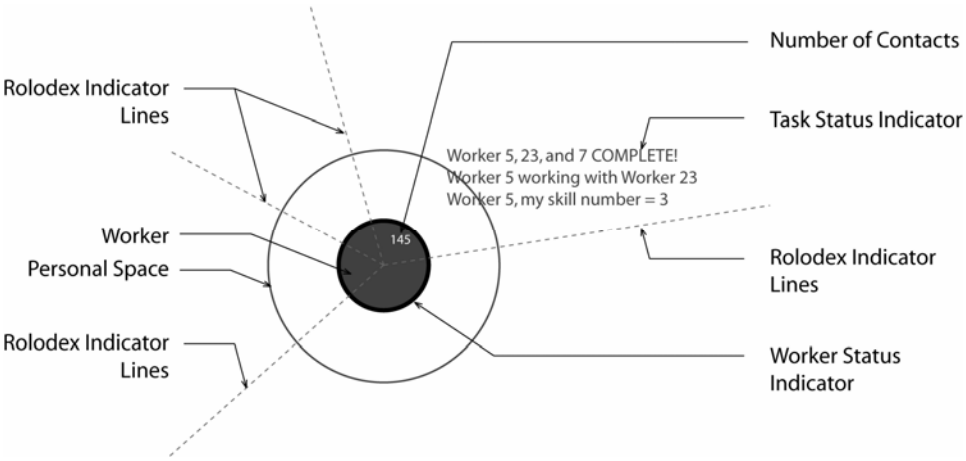

Fig. 1. Diagram of an individual worker

The office hierarchy is determined by each workers' cumulative number of contacts. (repeated contacts with the same worker are counted in this total). Considering a worker's contacts to be a measure of their popularity, the worker with the greatest number of contacts at any given moment is considered the Boss. All other rankings are based on the percentage of the Boss' total number of contacts. Workers of particulars rank are assigned to one of four smaller offices within the overall office environment. These smaller offices are randomly placed upon initialization of the simulation, but remain fixed thereafter, or until the simulation is ended and restarted (Fig. 2). The intention was to study whether changes in the location of the offices resulted in predictable patterns within the simulation. When initialized, the simulation assigns a set task to all the workers. Workers subsequently added to the system after initialization are also assigned the same task. The task requires the collaboration of at least three distinct workers to be completed.

These conditions attempt to create a system of autonomous objects whose global combinatorial complexity cannot realistically be predicted or anticipated. By eliminating the ability of an observer to anticipate the behavior of the simulation, the objective interpretation of patterns, or predictable cause-and-effect relationships can be explicated. Discrete states can subsequently be analyzed for patterns of behavior and/or organization by attempting to recreate those situations through repeated runs of the simulation. The practical advantage is that within a given timeframe, many more iterations of the simulation can be generated for comparative analysis than CAD drawings can be in the architectural plan.

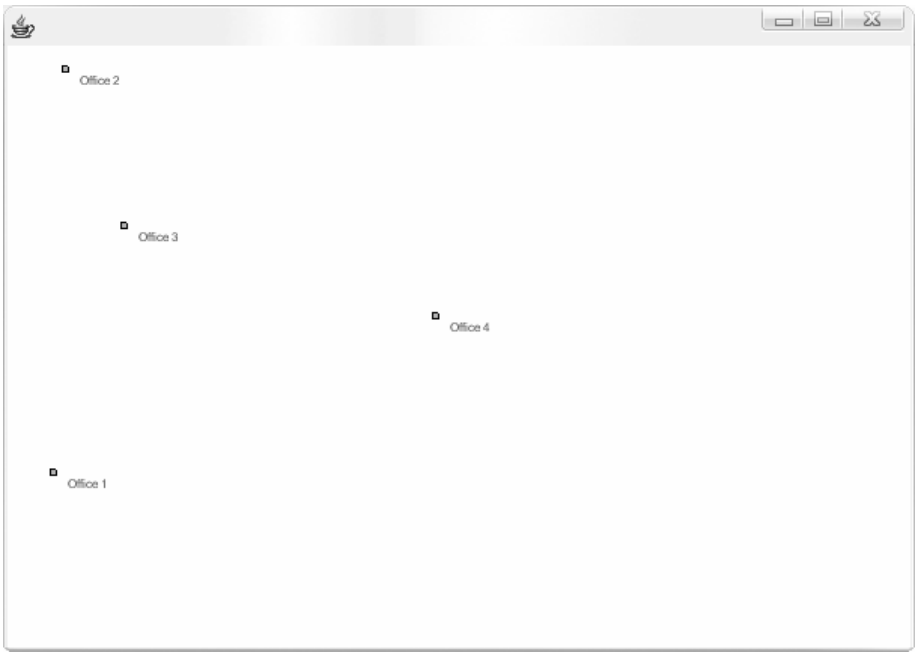

Fig. 2. Initial simulation layout.

3.1 The Use of Object-Oriented Programming

Abstracting a design idea into the formal framework of computer languages like *Processing*, *C#*, or *Java* is perhaps the most significant moment of divergence from other forms of conceptual representation. The desired operability of the system must be translated into data structures and logical functions within a semantically and syntactically-prescribed format. These constraints are what allows a design concept to be rigorously interrogated.

In the simulation, workers are defined as a class, or object, allowing the characteristics of each worker within the workforce to be commonly defined but independently manipulated. Rules of Euclidean space are employed to define functions which create object behaviors such as collision detection. Generic rules can be described as a function which can be repeated within other functions. In *theOffice* the same function which restricts workers from overlapping or moving beyond the office boundaries is also used to count and record the number of times workers come into contact with each other.

This number, called the ‘contact number’ can be stored within the program and compared to the contact numbers of every other worker. Taking the object with the most collisions to be the most active, and therefore best ‘well-known’, provides a method for establishing worker hierarchy. In this case, the worker with the most contacts is considered the Boss, with the highest ranking in the hierarchy. Because the number of contacts a worker has is constantly changing, the Boss’ position is continually in jeopardy of being lost to another more well-known worker, or one that accumulates more contacts. Conceptually, this creates a level of competition within the workforce. Consistent and irregular changes in hierarchy are good examples of the unpredictable and unanticipatable nature of the simulation.

3.2 Running the Simulation

When the simulation is initialized, the java applet window that is created is considered the extent of the office environment. Within the office, four small squares are randomly positioned which represent the four smaller offices. The user creates workers by clicking the mouse within the applet window – one worker is created per click at the position of pointer. Workers are represented by a solid colored circle of uniform size, which immediately begins moving about the office. The outline of a red circle surrounds the worker and indicates their sphere of influence. When introduced to the office a worker's influence is equivalent to the size of the worker, but over time the worker's influence will grow and shrink in accordance with the worker's proximal location to other workers. The overall space of a worker cannot exceed the boundaries of the office environment, it cannot overlap another worker's space, it can not be less than the size of the worker. This behavior is achieved by having each worker measure the distance between their center point and the center point of every other worker as well as the boundaries of the office environment (Fig. 3). For any two workers, if the actual distance between center points is equal to or less than the sum of the radii of the workers' overall space, the workers are considered to be in contact (Fig. 4) or overlapping (Fig. 5) and react to this condition by moving in a direction equal to the sum of their two direction vectors at a step equal to their degree of overlap.

The ranking hierarchy is determined by evaluating the number of contacts of each worker in the system. The worker with the most contacts at any given moment is the Boss, and all other rankings are evaluated as a percentage of the Boss' number of contacts. Workers in the 80-99th percentile are highlighted by a green line and their speed is increased. Workers in the 60-79th percentile are highlighted in blue and given an even greater increase in speed. Workers in the 40-59th percentile receive a dark blue highlight but no increase in speed, and workers below the 40th percentile remain unchanged. Each percentile from the 40th and up is assigned to one of the four randomly placed offices. As the dynamics of the overall workforce continue to evolve, an individual's rank is likely to change and thus their speed and the office to which they must report changes, furthering the overall workforce dynamics.

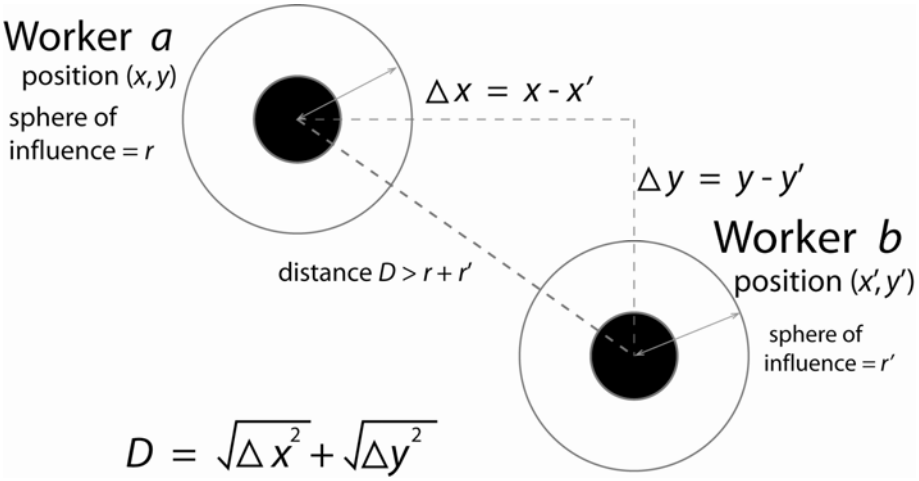

Fig. 3. Method used to determine agent collisions.

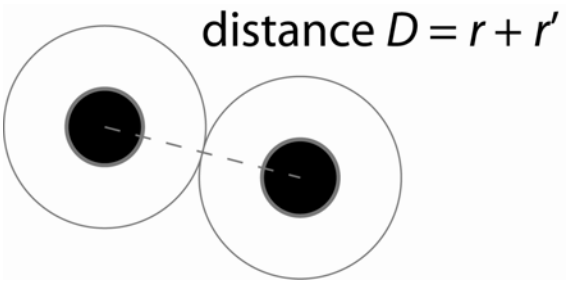

Fig. 4. Workers evaluated to be in contact.

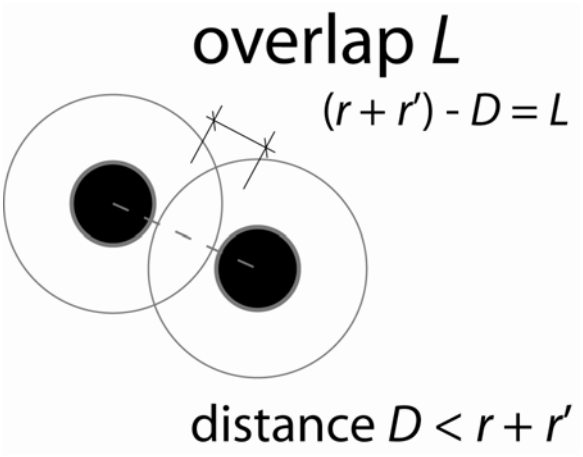

Fig. 5. Workers overlapping.

The ability to create collaboration among workers is achieved by assigning a global task to the workforce requiring the combined skills of multiple workers. Each worker is assigned a random, positive whole number between one and five which represents their 'skill'. The task itself is an array of three random, positive whole numbers from one to five. Upon creation, each worker contains an empty array which functions as a rolodex in which they can store the skill numbers of other workers they come into contact with. The rolodex stores the references of the last five unique contacts and a line is drawn from each worker to those in their rolodex. Using nested conditional statements, each worker searches through their rolodex to find other workers with the skills necessary to complete the task. When contacts with the proper workers have been made to satisfy this condition, the task is considered complete.

4 Results

The result of the project is a working simulation in which an abstract conceptual workforce interacts within an office environment. The dynamics of the system result in continuous changes in the hierarchy and spatial organization of the group. The user-interaction of the system allows new workers to be added at any given time, making experimentation with regard to the effect of new workers on the overall system possible.

Of note is the capacity for multiple workers to share a common hierarchal ranking. For example, if two or more workers both have the current maximum number of contacts, they are both considered to be the Boss. While this is a logical result of the way in which the hierarchy ranking is evaluated, it was an unintended behavior that was directly related to the conceptual understanding of office hierarchy employed. Iterations of *theOffice* that have been run have also displayed a tendency for a clustering of workers around office Two, the office to which workers within the 80-99th percentile are assigned. The increased speed of these workers causes them to cluster around the office and rapidly collide with each other. This has the effect of greatly increasing the contact numbers of workers within the cluster to the extent that the role of Boss changes every few seconds.

5 Conclusion

According to Donald Schon a design process is not one of problem solving, but one of problem setting [4]. This emphasizes the design question over the design solution – the setting of the design problem – as opposed to presupposing that the design is simply a formal outcome.

This investigation draws attention to various methods for analyzing, abstracting, and encoding conceptual theories in structured conditional and logical formats. The process is reflexive one, where every attribute of the system provides an opportunity for reflection and interrogation. The encoding process of writing a simulation embodies the critical moment during which intuition must take the form of logical Boolean conditions to be tested in a virtual space. Concepts of behavior must be abstracted, the simulated results of which must also be analyzed and interpreted to derive meaning – nothing remains assumed. The analysis and interpretation of cause and effect is the next critical moment during which the designer must consider the meaning of the simulation with respect to the design problem. If a change to the value of a single variable can dramatically alter the behavior of the entire system instantaneously, what does this mean in terms of its real-world counterpart?

Designers must carefully consider the way they choose to *see* design problems, and simulation provides a methodology for illuminating the seeing process. Revealing basic assumptions creates a space in which those assumptions may be interrogated, resulting in the potential for truly novel design solutions. If the end goal of a simulation undertaken is simply to see what happens, then that is all one will ever see. However, if a simulation is created with the intention of studying a particular scenario or design concept, and rules for that system are established based on explicit ideas about that concept, then the simulation becomes a valuable tool for understanding, analyzing, and critiquing design.

In conceptualizing and creating *theOffice*, intention played a crucial role in translating concepts of office behavior into the symbolic constructs of office workers and their methods of interacting. However, since there was no specified design problem or goal, it was not possible to determine the effectiveness of the simulation to stimulate unexpected or unanticipated design solutions. Also, it would be necessary to engage working professionals in the process to determine the effectiveness of writing code as an interface which is currently necessary to create simulations. Additional work in this area is required to assess the role of similar simulations in the design process.

There is little doubt that these or similar methodologies are likely to become more user-friendly in the near future. In her 1989 article, “The New Graphic Languages”

Muriel Cooper of the MIT Media Lab recounts the evolution of the creation of the graphic user interface which was necessary to allow designers to directly interface computers without the need for a separate “operator” [1]. This innovation did not make digital means necessary, but they created the opportunity for designers to explore and understand what role those new technologies could play in their work. Likewise, it will be necessary for a more designer-compatible interface before tools like simulation will become widely studied as alternative methods for design interrogation.

References

1. Cooper, M., (1989) “The New Graphic Languages” in *Design Quarterly* v.142
2. Kilian, A., (2006). Paraphrased from course lecture at MIT.
3. Minsky, M., (1986). *The Society of Mind*. New York: Simon and Schuster.
4. Schön, D., (1990). *Educating the Reflective Practitioner : Toward a New Design for Teaching and Learning in the Professions*. San Francisco: Jossey-Bass.
5. Scheurer, F., (2007) “Getting complexity organized: Using self-organisation in architectural construction” in *Automation in Construction* 16, p78-85.
6. Simon, H., (1996). *Sciences of the Artificial*. Third Edition: MIT Press.
7. Sutherland, I., (1975). "Structure in Drawings and the Hidden-Surface Problem." Chapter 7 in *Reflections on Computer Aids to Design and Architecture*. Edited by N. Negroponte. New York, NY: Petrocelli.